SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305-4022

A MATRIX FACTORIZATION AND ITS APPLICATION TO LARGE-SCALE LINEAR PROGRAMMING

by
Pierre F. de Mazancourt

TECHNICAL REPORT SOL 89-10

July 1989

Research and reproduction of this report were partially supported by the National Science Foundation grants ECS-8715153, DDM-8814253, DMS-8800589; U.S. Department of Energy grants DE-FG03-87ER25028, DE-FG03-87ER25030; and Office of Naval Research grant N00014-89-J-1659. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author and do NOT necessarily reflect the views of the above sponsors.

ABSTRACT

As an alternative to the LU matrix factorization, we consider a factorization that uses the lower triangular part of the original matrix as one factor and computes the other factors as a product of rank-one update matrices.

Under some non-singularity assumptions, an $m \times m$ matrix $\mathbf{A}$ can be factorized as $\mathbf{E}_m\mathbf{E}_{m-1}\ldots\mathbf{E}_2\mathbf{A}_1$ where $\mathbf{A}_1$ is the lower triangular part of $\mathbf{A}$ and $\mathbf{E}_k$ is a rank-one update matrix of the form $\mathbf{I} + \mathbf{v}_k\boldsymbol{\omega}_k$ with $\mathbf{v}_k$ a column vector and $\boldsymbol{\omega}_k$ a row vector. The vector $\mathbf{v}_k$ is the k^{th} column of $\mathbf{A} - \mathbf{A}_1$. If $\mathbf{v}_k = \mathbf{0}$, then $\mathbf{E}_k = \mathbf{I}$ may be omitted from the factorization. Otherwise, the row vector $\boldsymbol{\omega}_k$ must be computed.

After reviewing and improving the time complexity, the requirements, the stability and the efficiency of this method, we derive a stable factorization algorithm which we implement in FORTRAN77 within the framework of the simplex algorithm for linear programming.

A comparison of our numerical results with those obtained through the code MINOS 5.3 indicate that our method may be more efficient than an ordinary LU decomposition for some matrices whose order ranges between 28 and 1481, especially when these matrices are almost triangular.

ACKNOWLEDGEMENTS

I am very grateful to George B. Dantzig who showed confidence in me and who directed me during the early stages of my research, and to Michael A. Saunders who very patiently guided me through the last stages (and who also taught me a lot about editing!).

I am glad to have met Farid Ait-Sahlia, Robert Entriken, Joseph Mitchell, Francisco-Javier Prieto, Jennifer Rowley, John Stone and many graduate students whose help, advice or company I have enjoyed during my stay at Stanford.

I appreciate the administrative assistance which I have received from Sumi Kawasaki, Gail Stein and Audrey Stevenin.

This thesis has been written with hardware and software provided mostly by the Department of Operations Research at Stanford. The numerical computations have been run on a Sun workstation in the Department of Computer Science at Cornell.

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

The most widely used matrix factorization, the LU factorization, amounts to the computation of two triangular factors, one of which can be regarded as a product of rank-one update matrices. As an alternative, George B. Dantzig (1985) has proposed another factorization that uses the lower triangular part of the original matrix as one factor and computes the other factor as a product of rank-one update matrices.

Under some non-singularity assumptions, an $m \times m$ matrix $\mathbf{A}$ can be factorized as $\mathbf{E}_m \mathbf{E}_{m-1} \ldots \mathbf{E}_2 \mathbf{A}_1$ where $\mathbf{A}_1$ is the lower triangular part of $\mathbf{A}$ and $\mathbf{E}_k$ is a rank-one update matrix of the form $\mathbf{I} + \mathbf{v}_k \boldsymbol{\omega}_k$ with $\mathbf{v}_k$ a column vector and $\boldsymbol{\omega}_k$ a row vector.

The vector $\mathbf{v}_k$ is the k^{th} column of $\mathbf{A} - \mathbf{A}_1$. If $\mathbf{v}_k = \mathbf{0}$, then $\mathbf{E}_k = \mathbf{I}$ may be omitted from the factorization. Otherwise, the row vector $\boldsymbol{\omega}_k$ defining $\mathbf{E}_k$ can be obtained by solving $\boldsymbol{\omega}_k \mathbf{E}_{k-1} \ldots \mathbf{E}_2 \mathbf{A}_1 = \mathbf{u}_k^T$ where $\mathbf{u}_k^T$ is the k^{th} unit row vector.

Once the auxiliary vectors $\boldsymbol{\omega}_k$ have been computed, any system $\mathbf{Ax} = \mathbf{b}$ or $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ can be solved through one sparse triangular system involving $\mathbf{A}_1$ and s rank-one updates involving the matrices $\mathbf{E}_k$ for which column k is a spike (i.e. $\mathbf{v}_k \neq \mathbf{0}$).

However, that factorization may break down, for instance on a matrix whose diagonal contains a zero element. In addition, even if the factorization exists, it will often be unstable. In this thesis, we show how to overcome the problems of existence and stability. We present a factorization method theoretically applicable to any non-singular matrix. The numerical results presented in the last chapter indicate that our method may be more efficient than an ordinary LU factorization for some matrices whose order ranges from 28 to 1481, and whose number of elements ranges from 72 to 6344.

In Chapter **1**, we indicate how to compute the vectors $\boldsymbol{\omega}_k$ recursively, and how to solve systems such as $\mathbf{Ax} = \mathbf{b}$ and $\pi\mathbf{A} = \boldsymbol{\gamma}$ using these vectors, assuming the non-singularity of **A** and of some of its submatrices.

In Chapter **2**, we streamline the method described in Chapter **1** and reduce its computational complexity to the same level as that of the LU factorization.

In Chapter **3**, we study the existence and the stability of the factorization. We introduce the vectors $\boldsymbol{\sigma}_k$ $(2 \leq k \leq m)$, a normalized form of the vectors $\boldsymbol{\omega}_k$ $(2 \leq k \leq m)$. If **A** is non-singular, a permutation **Q** of the columns of **A** makes **AQ** and its leading principal submatrices non-singular and, in practice, well conditioned. We indicate how to compute simultaneously the permutation **Q** and the vectors $\boldsymbol{\sigma}_k$ representing the factorization of **AQ**.

In Chapter **4**, we present some algorithms inspired by Hellerman and Rarick (1971), that permute the rows and columns of **A** in order to reduce the number of spikes (i.e. the columns that have nonzero entries above the diagonal). The method is more efficient if there are fewer spikes and if the spikes are shifted towards the left because $\boldsymbol{\omega}_k$ has at most k nonzero components.

In Chapter **5**, we propose an algorithm that reorders the rows and columns of **A** while computing the vectors $\boldsymbol{\sigma}_k$, so as to reduce the number of spikes without compromising the numerical stability of the factorization.

In Chapter **6**, we describe a FORTRAN implementation of the algorithm within the framework of linear programming. We use our own set of factorization routines in the optimization code MINOS 5.3 of Murtagh and Saunders (1987) on a set of 51 test problems from *netlib* (Gay, 1985). With the options chosen, our method achieves faster running times than the original MINOS code on about a third of the problems. Unfortunately, it also takes up to four times as long as MINOS on some problems.

CHAPTER 1. THE BASIC METHOD

1.1 Introduction

Under some non-singularity assumptions, an $m \times m$ matrix $\mathbf{A}$ can be factorized into $\mathbf{E}_m\mathbf{E}_{m-1}\ldots\mathbf{E}_2\mathbf{A}_1$ where $\mathbf{A}_1$ is the lower triangular part of $\mathbf{A}$ and $\mathbf{E}_k$ is a rank-one update matrix of the form $\mathbf{I} + \mathbf{v}_k\boldsymbol{\omega}_k$ with $\mathbf{v}_k$ a column vector and $\boldsymbol{\omega}_k$ a row vector.

The vector $\mathbf{v}_k$ is the k^{th} column of $\mathbf{A} - \mathbf{A}_1$. If $\mathbf{v}_k = \mathbf{0}$, then $\mathbf{E}_k = \mathbf{I}$ may be omitted from the factorization. Otherwise, the row vector $\boldsymbol{\omega}_k$ must be computed.

In this Chapter, we describe the factorization and provide a natural method to compute the vectors $\boldsymbol{\omega}_k$ $(2 \leq k \leq m)$ and to solve systems such as $\mathbf{Ax} = \mathbf{b}$ or $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ using this factorization.

1.2 Definitions and Notation

Let m be a positive integer. Let $\mathbf{R}$ denote the real numbers. Unless otherwise specified, matrix denotes an element of $\mathbf{R}^{m\times m}$, column vector denotes an element of $\mathbf{R}^{m\times 1}$ and row vector denotes an element of $\mathbf{R}^{1\times m}$. Row vectors are usually represented by Greek letters.

A given matrix $\mathbf{A}$ can be decomposed into its lower triangular part $\mathbf{A}_1$ (including the diagonal) and its strictly upper triangular part, which can in turn be broken down according to its entries in columns $2,\ldots,m$ into $m-1$ matrices of the form $\mathbf{v}_k\mathbf{u}_k^T$ for $2 \leq k \leq m$, where $\mathbf{u}_k$ is the k^{th} unit vector. Then

$$\mathbf{A} \;=\; \mathbf{A}_1 \;+\; \mathbf{v}_2\mathbf{u}_2^T \;+\; \mathbf{v}_3\mathbf{u}_3^T \;+\; \cdots \;+\; \mathbf{v}_m\mathbf{u}_m^T.$$

For $2 \leq k \leq m$, let

$$\begin{aligned}\mathbf{A}_k &\overset{\Delta}{=} \mathbf{A}_{k-1} \;+\; \mathbf{v}_k\mathbf{u}_k^T \\ &= \mathbf{A}_1 \;+\; \mathbf{v}_2\mathbf{u}_2^T \;+\; \cdots \;+\; \mathbf{v}_k\mathbf{u}_k^T.\end{aligned}$$

If $\mathbf{v}_k \neq \mathbf{0}$, column k is called a spike of $\mathbf{A}$.

Example: The following 3×3 matrix $\mathbf{A}$ has two spikes, namely columns 2 and 3. We use th ymbol $*$ to denote coefficients that are identically zero.

$$\begin{aligned}
\mathbf{A} &= \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \\
&= \begin{pmatrix} 8 & * & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} + \begin{pmatrix} * & 1 & * \\ * & * & * \\ * & * & * \end{pmatrix} + \begin{pmatrix} * & * & 6 \\ * & * & 7 \\ * & * & * \end{pmatrix}
\end{aligned}$$

$$\mathbf{A}_1 = \begin{pmatrix} 8 & * & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} \qquad \mathbf{v}_2 = \begin{pmatrix} 1 \\ * \\ * \end{pmatrix} \qquad \mathbf{u}_2^T = (\, * \quad 1 \quad * \,)$$

$$\mathbf{A}_2 = \begin{pmatrix} 8 & 1 & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} \qquad \mathbf{v}_3 = \begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix} \qquad \mathbf{u}_3^T = (\, * \quad * \quad 1 \,)$$

$$\mathbf{A}_3 = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$$

In this chapter, we assume that **the matrices $\mathbf{A}_k$ $(1 \leq k \leq m)$ are nonsingular** (cf. Chapter **3**). We can then turn the above decomposition of $\mathbf{A}$ into a factorization. For $2 \leq k \leq m$, let

$$\begin{cases} \boldsymbol{\omega}_k \triangleq \mathbf{u}_k^T \mathbf{A}_{k-1}^{-1} \\ \mathbf{E}_k \triangleq \mathbf{I} + \mathbf{v}_k \boldsymbol{\omega}_k \end{cases}$$

Then

$$\begin{aligned}
\mathbf{A}_k &= \mathbf{A}_{k-1} + \mathbf{v}_k \mathbf{u}_k^T \\
&= (\mathbf{I} + \mathbf{v}_k \mathbf{u}_k^T \mathbf{A}_{k-1}^{-1})\ \mathbf{A}_{k-1} \\
&= (\mathbf{I} + \mathbf{v}_k \boldsymbol{\omega}_k)\ \mathbf{A}_{k-1} \\
&= \mathbf{E}_k \mathbf{A}_{k-1} \\
&= \mathbf{E}_k \mathbf{E}_{k-1} \ldots \mathbf{E}_2 \mathbf{A}_1
\end{aligned}$$

and in particular, $\mathbf{A} = \mathbf{E}_m \mathbf{E}_{m-1} \ldots \mathbf{E}_2 \mathbf{A}_1$. Each matrix $\mathbf{E}_k$ is a rank-one update matrix. Note that we could also have factored $\mathbf{A}_1$ on the left and obtained a factorization of the form $\mathbf{A}_1 \mathbf{F}_2 \ldots \mathbf{F}_{k-1} \mathbf{F}_k$ where each matrix $\mathbf{F}_k$ is also a rank-one update matrix.

1.3 Solution of Rank-One Update Systems

Let $\mathbf{E} \triangleq \mathbf{I} + \mathbf{v}\boldsymbol{\omega}$ and $\theta \triangleq 1 + \boldsymbol{\omega}\mathbf{v}$. The following Lemmas are well known properties of rank-one update matrices (Golub & Van Loan, 1983).

Lemma 1.3.1

$$\begin{cases} \theta = \det \mathbf{E} \\ \mathbf{E}^{-1} = \mathbf{I} - \theta^{-1}\mathbf{v}\boldsymbol{\omega} \end{cases} .$$

Lemma 1.3.2 The solutions to the elementary linear systems $\mathbf{E}\mathbf{x} = \mathbf{b}$ and $\boldsymbol{\pi}\mathbf{E} = \boldsymbol{\gamma}$ are given by the following rank-one update formulas:

$$\begin{cases} \theta = 1 + \boldsymbol{\omega}\mathbf{v} \\ \mathbf{x} = \mathbf{b} - \theta^{-1}(\boldsymbol{\omega}\mathbf{b})\,\mathbf{v} \\ \boldsymbol{\pi} = \boldsymbol{\gamma} - \theta^{-1}(\boldsymbol{\gamma}\mathbf{v})\,\boldsymbol{\omega} \end{cases} .$$

Once $\boldsymbol{\omega}$ and θ have been computed, the number of multiplications or divisions required in either update is at most equal to the number of nonzero components in $\mathbf{v}$, plus the number of nonzero components in $\boldsymbol{\omega}$, plus one (corresponding to the factor θ^{-1}). By construction, the vectors $\mathbf{v}_k$ and $\boldsymbol{\omega}_k$ defined in Section **1.2** have at most $k-1$ and k nonzero components respectively. Therefore, solving the system $\mathbf{E}_k\mathbf{x} = \mathbf{b}$ or $\boldsymbol{\pi}\mathbf{E}_k = \boldsymbol{\gamma}$ requires at most $2k$ multiplications or divisions. The same upper bound holds for the number of additions or subtractions.

1.4 Computation of the Factors $\mathbf{E}_k$ $(2 \le k \le m)$

For our purposes, knowing the factor $\mathbf{E}_k = \mathbf{I} + \mathbf{v}_k\boldsymbol{\omega}_k$ through .he vectors $\mathbf{v}_k$ and $\boldsymbol{\omega}_k$ is sufficient. Since the vectors $\mathbf{v}_k$ are given, we need only compute the auxiliary vectors $\boldsymbol{\omega}_k$ $(2 \le k \le m)$ defined by $\boldsymbol{\omega}_k\mathbf{A}_{k-1} = \mathbf{u}_k^T$, i.e.

$\boldsymbol{\omega}_k \mathbf{E}_{k-1} \ldots \mathbf{E}_2 \mathbf{A}_1 = \mathbf{u}_k^T$. This can be done recursively by solving the systems

$$\begin{cases} \boldsymbol{\omega}_2 \mathbf{A}_1 = \mathbf{u}_2^T \\ \boldsymbol{\omega}_3 \mathbf{E}_2 \mathbf{A}_1 = \mathbf{u}_3^T \\ \vdots \\ \boldsymbol{\omega}_k \mathbf{E}_{k-1} \ldots \mathbf{E}_2 \mathbf{A}_1 = \mathbf{u}_k^T \\ \vdots \\ \boldsymbol{\omega}_m \mathbf{E}_{m-1} \mathbf{E}_{m-2} \ldots \mathbf{E}_2 \mathbf{A}_1 = \mathbf{u}_m^T \end{cases}$$

in that order. To solve the system $\boldsymbol{\omega}_k \mathbf{E}_{k-1} \ldots \mathbf{E}_2 \mathbf{A}_1 = \mathbf{u}_k^T$, we can solve the triangular system $\boldsymbol{\pi}_1 \mathbf{A}_1 = \mathbf{u}_k^T$ by backward substitution and then successively solve $k-2$ rank-one update systems of the form $\boldsymbol{\pi}_l \mathbf{E}_l = \boldsymbol{\pi}_{l-1}$ for $2 \leq l \leq k-1$. By Lemma **1.3.2**, these systems are equivalent to

$$\boldsymbol{\pi}_l = \boldsymbol{\pi}_{l-1} - \theta_l^{-1} (\boldsymbol{\pi}_{l-1} \mathbf{v}_l) \boldsymbol{\omega}_l \qquad \text{for} \quad 2 \leq l \leq k-1.$$

Once $\boldsymbol{\omega}_k = \boldsymbol{\pi}_{k-1}$ is known, we can obtain θ_k by

$$\theta_k = 1 + \boldsymbol{\omega}_k \mathbf{v}_k.$$

Since the last $m-k$ components of $\boldsymbol{\pi}_1$ and its successive rank-one updates (including $\boldsymbol{\omega}_k$) are zero, all these systems are of dimension at most k for practical purposes. The maximum number of multiplications required to compute $(\boldsymbol{\omega}_k, \theta_k)$ by this method is

$$\tfrac{1}{2}k(k+1) + \sum_{l=2}^{k-1} 2l + k - 1 \approx \frac{3}{2}k^2 + k$$

and the maximum number of multiplications required to factorize $\mathbf{A}$ is

$$\sum_{k=2}^{m} \left(\tfrac{3}{2}k^2 + k\right) \approx \tfrac{1}{2}m^3 + m^2.$$

Example:

Consider the 3×3 matrix $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$ introduced in Section **1.2**.

The vector $\boldsymbol{\omega}_2$ characterizing $\mathbf{E}_2$ is given by $\boldsymbol{\omega}_2\mathbf{A}_1 = \mathbf{u}_2^T$.

$$\boldsymbol{\omega}_2 \begin{pmatrix} 8 & * & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} = (\,0 \quad 1 \quad 0\,)$$

$$\boldsymbol{\omega}_2 \;=\; (\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,)$$

$$\theta_2 \;=\; 1 \;+\; (\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,)\begin{pmatrix} 1 \\ * \\ * \end{pmatrix} \;=\; \tfrac{37}{40}$$

The vector $\boldsymbol{\omega}_3$ characterizing $\mathbf{E}_3$ is given by $\boldsymbol{\omega}_3\mathbf{E}_2\mathbf{A}_1 = \mathbf{u}_3^T$.

$$\boldsymbol{\omega}_3\mathbf{E}_2 \begin{pmatrix} 8 & * & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} = (\,0 \quad 0 \quad 1\,)$$

$$\boldsymbol{\omega}_3\mathbf{E}_2 = (\,-\tfrac{3}{40} \quad -\tfrac{9}{10} \quad \tfrac{1}{2}\,)$$

$$\boldsymbol{\omega}_3 = (\,-\tfrac{3}{40} \quad -\tfrac{9}{10} \quad \tfrac{1}{2}\,) \;-\; \tfrac{40}{37}\left[(\,-\tfrac{3}{40} \quad -\tfrac{9}{10} \quad \tfrac{1}{2}\,)\begin{pmatrix} 1 \\ * \\ * \end{pmatrix}\right](\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,)$$

$$\boldsymbol{\omega}_3 = (\,\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}\,)$$

$$\theta_3 \;=\; 1 \;+\; (\,\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}\,)\begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix} \;=\; -\tfrac{180}{37}$$

1.5 Solution of $\mathbf{Ax} = \mathbf{b}$

To solve the system $\mathbf{Ax} = \mathbf{b}$ given the non-singular factorization $\mathbf{A} = \mathbf{E}_m \ldots \mathbf{E}_2\mathbf{A}_1$, the natural method is to solve successively the systems

$$\begin{cases} \mathbf{E}_m\mathbf{b}_{m-1} = \mathbf{b}_m \\ \qquad \vdots \\ \mathbf{E}_k\mathbf{b}_{k-1} = \mathbf{b}_k \\ \qquad \vdots \\ \mathbf{E}_2\mathbf{b}_1 = \mathbf{b}_2 \\ \mathbf{A}_1\mathbf{x} = \mathbf{b}_1 \end{cases}$$

for $\mathbf{b}_{m-1}, \ldots, \mathbf{b}_1$ and $\mathbf{x}$, having let $\mathbf{b}_m \triangleq \mathbf{b}$. Applying Lemma **1.3.2** to the non-singular matrices $\mathbf{E}_k$ $(m \geq k \geq 2)$, we can solve the first $m-1$ systems according to the rank-one update formulas

$$\mathbf{b}_{k-1} = \mathbf{b}_k - \theta_k^{-1} \left(\boldsymbol{\omega}_k \mathbf{b}_k \right) \mathbf{v}_k \qquad \text{for} \quad m \geq k \geq 2$$

and then solve the triangular system $\mathbf{A}_1 \mathbf{x} = \mathbf{b}_1$ by forward substitution.

The maximum number of multiplications required by this method is

$$\sum_{k=m}^{2} 2k + \tfrac{1}{2} m(m+1) \approx \tfrac{3}{2} m^2 + \tfrac{3}{2} m.$$

Example:

Consider the 3×3 system $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix}$. The system can be written $\mathbf{E}_3 \mathbf{E}_2 \mathbf{A}_1 \mathbf{x} = \mathbf{b}$. The vectors $\mathbf{b}_3$, $\mathbf{b}_2$, $\mathbf{b}_1$ and $\mathbf{x}$ are the following:

$$\mathbf{b}_3 = \begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix}$$

$$\mathbf{b}_2 = \begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix} + \tfrac{37}{180} \left[\left(\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2} \right) \begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix} \right] \begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix} = \begin{pmatrix} 6 \\ -7 \\ -8 \end{pmatrix}$$

$$\mathbf{b}_1 = \begin{pmatrix} 6 \\ -7 \\ -8 \end{pmatrix} + \tfrac{40}{37} \left[\left(-\tfrac{3}{40} \quad \tfrac{1}{5} \quad * \right) \begin{pmatrix} 6 \\ -7 \\ -8 \end{pmatrix} \right] \begin{pmatrix} 1 \\ * \\ * \end{pmatrix} = \begin{pmatrix} 8 \\ -7 \\ -8 \end{pmatrix}$$

$$\begin{pmatrix} 8 & * & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} \mathbf{x} = \begin{pmatrix} 8 \\ -7 \\ -8 \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}$$

1.6 Solution of $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$

To solve the system $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ given the non-singular factorization $\mathbf{A} = \mathbf{E}_m \ldots \mathbf{E}_2\mathbf{A}_1$, the natural method is to solve successively the systems

$$\begin{cases} \boldsymbol{\pi}_1\mathbf{A}_1 = \boldsymbol{\gamma} \\ \boldsymbol{\pi}_2\mathbf{E}_2 = \boldsymbol{\pi}_1 \\ \quad\vdots \\ \boldsymbol{\pi}_k\mathbf{E}_k = \boldsymbol{\pi}_{k-1} \\ \quad\vdots \\ \boldsymbol{\pi}_m\mathbf{E}_m = \boldsymbol{\pi}_{m-1} \end{cases}$$

for $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_m = \boldsymbol{\pi}$. The triangular system $\boldsymbol{\pi}_1\mathbf{A}_1 = \boldsymbol{\gamma}$ can be solved by backward substitution. Then, applying Lemma **1.3.2** to the non-singular matrices $\mathbf{E}_k$, we can solve the last $m-1$ systems according to the rank-one update formulas

$$\boldsymbol{\pi}_k \;=\; \boldsymbol{\pi}_{k-1} - \;\theta_k^{-1}\,(\boldsymbol{\pi}_{k-1}\mathbf{v}_k)\,\boldsymbol{\omega}_k \qquad \text{for} \quad 2 \le k \le m.$$

The maximum number of multiplications required by this method is

$$\tfrac{1}{2}m(m+1) \;+\; \sum_{k=2}^{m} 2k \;\approx\; \tfrac{3}{2}m^2 + \tfrac{3}{2}m.$$

Example:

Consider the 3×3 system $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ where $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$ and $\boldsymbol{\gamma} = (\,6 \quad 0 \quad -6\,)$. The system can be written $\boldsymbol{\pi}\mathbf{E}_3\mathbf{E}_2\mathbf{A}_1 = \boldsymbol{\gamma}$. The vectors $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$, $\boldsymbol{\pi}_3$ and $\boldsymbol{\pi}$ are the following:

$$\boldsymbol{\pi}_1 \begin{pmatrix} 8 & * & * \\ 3 & 5 & * \\ 4 & 9 & 2 \end{pmatrix} = (\,6 \quad 0 \quad -6\,)$$

$$\boldsymbol{\pi}_1 \;=\; (\,\tfrac{9}{40} \quad \tfrac{27}{5} \quad -3\,)$$

$$\begin{aligned}
\boldsymbol{\pi}_2 &= \begin{pmatrix} \frac{9}{40} & \frac{27}{5} & -3 \end{pmatrix} - \frac{40}{37}\left[\begin{pmatrix} \frac{9}{40} & \frac{27}{5} & -3 \end{pmatrix}\begin{pmatrix} 1 \\ * \\ * \end{pmatrix}\right]\begin{pmatrix} -\frac{3}{40} & \frac{1}{5} & * \end{pmatrix} \\
&= \begin{pmatrix} \frac{9}{37} & \frac{198}{37} & -3 \end{pmatrix} \\
\boldsymbol{\pi}_3 &= \begin{pmatrix} \frac{9}{37} & \frac{198}{37} & -3 \end{pmatrix} + \frac{37}{180}\left[\begin{pmatrix} \frac{9}{37} & \frac{198}{37} & -3 \end{pmatrix}\begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix}\right]\begin{pmatrix} \frac{7}{74} & -\frac{34}{37} & \frac{1}{2} \end{pmatrix} \\
&= \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}
\end{aligned}$$

$$\boldsymbol{\pi} = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$$

1.7 Fundamental Observation

If column l is not a spike (i.e. if $\mathbf{v}_l = 0$), then $\mathbf{E}_l = \mathbf{I}$ may be omitted from the factorization of $\mathbf{A}$ and the rank-one update corresponding to $\mathbf{E}_l$ may be skipped when solving $\mathbf{Ax} = \mathbf{b}$, $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ or even $\boldsymbol{\omega}_k\mathbf{A}_{k-1} = \mathbf{u}_k^T$. This simplification significantly reduces the size of the computations when the matrix $\mathbf{A}$ is large and has few spikes.

Example:

This example shows how to factorize a 5 × 5 matrix whose columns 2 and 4 are not spikes, and how to solve systems like $\mathbf{Ax} = \mathbf{b}$ or $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 6 \\ 1 \\ 6 \\ 6 \\ 8 \end{pmatrix} \qquad \boldsymbol{\gamma} = \begin{pmatrix} 1 & 3 & 3 & 2 & 3 \end{pmatrix}$$

$$\mathbf{A}_1 = \begin{pmatrix} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ -1 & 0 & 1 & * & * \\ 0 & 1 & 1 & 1 & * \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{A}_2 = \mathbf{A}_1$$

$$\mathbf{A}_3 = \begin{pmatrix} 1 & 0 & 1 & * & * \\ 0 & 1 & 0 & * & * \\ -1 & 0 & 1 & * & * \\ 0 & 1 & 1 & 1 & * \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{A}_4 = \mathbf{A}_3$$

$$\mathbf{A}_5 = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{v}_2 = \mathbf{0} \quad \Longrightarrow \quad \mathbf{E}_2 = \mathbf{I}$$

$$\mathbf{v}_3 = \begin{pmatrix} 1 \\ 0 \\ * \\ * \\ * \end{pmatrix}$$

$$\mathbf{v}_4 = \mathbf{0} \quad \Longrightarrow \quad \mathbf{E}_4 = \mathbf{I}$$

$$\mathbf{v}_5 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ * \end{pmatrix}$$

- The vector $\boldsymbol{\omega}_3$ characterizing $\mathbf{E}_3$ is given by $\boldsymbol{\omega}_3\mathbf{A}_1 = \mathbf{u}_3^T$.

$$\boldsymbol{\omega}_3 \begin{pmatrix} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ -1 & 0 & 1 & * & * \\ 0 & 1 & 1 & 1 & * \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{\omega}_3 = \begin{pmatrix} 1 & 0 & 1 & * & * \end{pmatrix}$$

$$\boldsymbol{\theta}_3 = 1 + \begin{pmatrix} 1 & 0 & 1 & * & * \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ * \\ * \\ * \end{pmatrix} = 2$$

- The vector $\boldsymbol{\omega}_5$ characterizing $\mathbf{E}_5$ is given by $\boldsymbol{\omega}_5\mathbf{E}_3\mathbf{A}_1 = \mathbf{u}_5^T$.

$$\boldsymbol{\omega}_5\mathbf{E}_3 \begin{pmatrix} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ -1 & 0 & 1 & * & * \\ 0 & 1 & 1 & 1 & * \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\omega_5\mathbf{E}_3 = (0 \quad 0 \quad 1 \quad -1 \quad 1)$$

$$\omega_5 = (0 \quad 0 \quad 1 \quad -1 \quad 1) - \tfrac{1}{2}\left[(0 \quad 0 \quad 1 \quad -1 \quad 1)\begin{pmatrix}1\\0\\*\\*\\*\end{pmatrix}\right](1 \quad 0 \quad 1 \quad * \quad *)$$

$$\omega_5 = (0 \quad 0 \quad 1 \quad -1 \quad 1)$$

$$\theta_5 = 1 + (0 \quad 0 \quad 1 \quad -1 \quad 1)\begin{pmatrix}1\\0\\1\\0\\*\end{pmatrix} = 2$$

- The system $\mathbf{Ax} = \mathbf{b}$ can be written $\mathbf{E}_5\mathbf{E}_3\mathbf{A}_1\mathbf{x} = \mathbf{b}$. The vectors $\mathbf{b}_5$, $\mathbf{b}_3$, $\mathbf{b}_1$ and $\mathbf{x}$ are the following:

$$\mathbf{b}_5 = \begin{pmatrix}6\\1\\6\\6\\8\end{pmatrix}$$

$$\mathbf{b}_3 = \begin{pmatrix}6\\1\\6\\6\\8\end{pmatrix} - \tfrac{1}{2}\left[(0 \quad 0 \quad 1 \quad -1 \quad 1)\begin{pmatrix}6\\1\\6\\6\\8\end{pmatrix}\right]\begin{pmatrix}1\\0\\1\\0\\*\end{pmatrix} = \begin{pmatrix}2\\1\\2\\6\\8\end{pmatrix}$$

$$\mathbf{b}_1 = \begin{pmatrix}2\\1\\2\\6\\8\end{pmatrix} - \tfrac{1}{2}\left[(1 \quad 0 \quad 1 \quad * \quad *)\begin{pmatrix}2\\1\\2\\6\\8\end{pmatrix}\right]\begin{pmatrix}1\\0\\*\\*\\*\end{pmatrix} = \begin{pmatrix}0\\1\\2\\6\\8\end{pmatrix}$$

$$\begin{pmatrix}1 & * & * & * & *\\0 & 1 & * & * & *\\-1 & 0 & 1 & * & *\\0 & 1 & 1 & 1 & *\\1 & 1 & 0 & 1 & 1\end{pmatrix}\mathbf{x} = \begin{pmatrix}0\\1\\2\\6\\8\end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix}0\\1\\2\\3\\4\end{pmatrix}$$

- The system $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ can be written $\boldsymbol{\pi}\mathbf{E}_5\mathbf{E}_3\mathbf{A}_1 = \boldsymbol{\gamma}$. The vectors $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_3$, $\boldsymbol{\pi}_5$ and $\boldsymbol{\pi}$ are the following:

$$\boldsymbol{\pi}_1 \begin{pmatrix} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ -1 & 0 & 1 & * & * \\ 0 & 1 & 1 & 1 & * \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} = (1 \quad 3 \quad 3 \quad 2 \quad 3)$$

$$\boldsymbol{\pi}_1 = (2 \quad 1 \quad 4 \quad -1 \quad 3)$$

$$\begin{aligned} \boldsymbol{\pi}_3 &= (2 \quad 1 \quad 4 \quad -1 \quad 3) - \tfrac{1}{2}\left[(2 \quad 1 \quad 4 \quad -1 \quad 3)\begin{pmatrix} 1 \\ 0 \\ * \\ * \\ * \end{pmatrix}\right](1 \quad 0 \quad 1 \quad * \quad *) \\ &= (1 \quad 1 \quad 3 \quad -1 \quad 3) \\ \boldsymbol{\pi}_5 &= (1 \quad 1 \quad 3 \quad -1 \quad 3) - \tfrac{1}{2}\left[(1 \quad 1 \quad 3 \quad -1 \quad 3)\begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ * \end{pmatrix}\right](0 \quad 0 \quad 1 \quad -1 \quad 1) \\ &= (1 \quad 1 \quad 1 \quad 1 \quad 1) \end{aligned}$$

$$\boldsymbol{\pi} = (1 \quad 1 \quad 1 \quad 1 \quad 1)$$

CHAPTER 2. THE STREAMLINED METHOD

2.1 Introduction

The basic method of Chapter 1 is not the most efficient way to compute and use the factorization $\mathbf{A} = \mathbf{E}_m\mathbf{E}_{m-1}\ldots\mathbf{E}_2\mathbf{A}_1$. In this Chapter, we present a streamlined method that yields the same results as the basic method without explicitly solving the intermediate system involving the triangular matrix $\mathbf{A}_1$. This decreases by about 33% the upper bound on the number of multiplications required to compute the factorization of $\mathbf{A}$, the solution of $\mathbf{Ax} = \mathbf{b}$ or the solution of $\pi\mathbf{A} = \gamma$.

2.2 Solution of $\mathbf{Ax} = \mathbf{b}$

The basic method explained in Section **1.5** can be streamlined by observing that x_k, the k^{th} component of $\mathbf{x}$, is computed when the rank-one update system $\mathbf{E}_k\mathbf{b}_{k-1} = \mathbf{b}_k$ is solved. Consider the following Lemmas:

Lemma 2.2.1 For $2 \le k \le m$, $\mathbf{u}_k^T = \theta_k^{-1}\omega_k\mathbf{A}_k$.

Proof θ_k, the determinant of $\mathbf{E}_k$, is nonzero and

$$\begin{aligned}
\theta_k\mathbf{u}_k^T &= (1+\omega_k\mathbf{v}_k)\,\mathbf{u}_k^T \\
&= \mathbf{u}_k^T + \omega_k\mathbf{v}_k\mathbf{u}_k^T \\
&= \omega_k\mathbf{A}_{k-1} + \omega_k\mathbf{v}_k\mathbf{u}_k^T \\
&= \omega_k\left(\mathbf{A}_{k-1} + \mathbf{v}_k\mathbf{u}_k^T\right) \\
&= \omega_k\mathbf{A}_k.
\end{aligned}$$

Lemma 2.2.2 For $2 \le k \le m$, $\mathbf{A}_k\mathbf{x} = \mathbf{b}_k$.

Proof By definition of $\mathbf{b}_k$ and $\mathbf{A}_k$, we have

$$\begin{aligned} \mathbf{b}_k &= \mathbf{E}_k\mathbf{b}_{k-1} \\ &= \mathbf{E}_k\mathbf{E}_{k-1}\mathbf{b}_{k-2} \\ &= \mathbf{E}_k\mathbf{E}_{k-1}\ldots\mathbf{E}_2\mathbf{A}_1\mathbf{x} \\ &= \mathbf{A}_k\mathbf{x}. \end{aligned}$$

Lemma 2.2.3 For $2 \le k \le m, \quad x_k = \theta_k^{-1}\boldsymbol{\omega}_k\mathbf{b}_k$.

Proof By Lemmas **2.2.1** and **2.2.2**,

$$\begin{aligned} x_k &= \mathbf{u}_k^T\mathbf{x} \\ &= \theta_k^{-1}\boldsymbol{\omega}_k\mathbf{A}_k\mathbf{x} \\ &= \theta_k^{-1}\boldsymbol{\omega}_k\mathbf{b}_k. \end{aligned}$$

Lemma 2.2.4 For $2 \le k \le m, \quad \mathbf{b}_{k-1} = \mathbf{b}_k - x_k\,\mathbf{v}_k$.

Proof Starting with Lemma **2.2.2**, we obtain

$$\begin{aligned} \mathbf{b}_{k-1} &= \mathbf{A}_{k-1}\mathbf{x} \\ &= (\mathbf{A}_k - \mathbf{v}_k\mathbf{u}_k^T)\,\mathbf{x} \\ &= \mathbf{A}_k\mathbf{x} - \mathbf{v}_k\mathbf{u}_k^T\mathbf{x} \\ &= \mathbf{b}_k - \mathbf{v}_k x_k. \end{aligned}$$

Lemmas **2.2.3** and **2.2.4** provide the same update formula as Section **1.5** but they show that the components x_k $(m \ge k \ge 2)$ can be computed along with the vectors $\mathbf{b}_{k-1}$ $(m \ge k \ge 2)$ without additional work. By the time $\mathbf{b}_1$ is computed, the only component of $\mathbf{x}$ that remains unknown is x_1. At that stage, instead of solving the whole triangular system $\mathbf{A}_1\mathbf{x} = \mathbf{b}_1$, we only need to solve the first equation, of the form $A_{11}x_1 = \mathbf{u}_1^T\mathbf{b}_1$. In summary, we obtain the following method.

Algorithm 2.2 (to solve $\mathbf{Ax} = \mathbf{b}$)

Let $\mathbf{b}_m = \mathbf{b}$.

For $k = m$ downto 2, let $x_k = \theta_k^{-1}\boldsymbol{\omega}_k\mathbf{b}_k$ and $\mathbf{b}_{k-1} = \mathbf{b}_k - x_k\,\mathbf{v}_k$.

Let $x_1 = A_{11}^{-1}\mathbf{u}_1^T\mathbf{b}_1$.

This algorithm actually depends on the auxiliary vectors $A_{11}^{-1}\,\mathbf{u}_1^T$ and $\theta_k^{-1}\omega_k$ $(2 \leq k \leq m)$ which are none other than the vectors $\mathbf{u}_k^T\mathbf{A}_k^{-1}$ $(1 \leq k \leq m)$ (cf. Lemma **2.2.1**). The maximum number of multiplications required by this method is

$$\sum_{k=m}^{2} 2k \quad + \ 1 \ \approx \ m^2 + m.$$

This is essentially the same as the maximum number of multiplications required to solve a system $\mathbf{LUx} = \mathbf{b}$ where $\mathbf{L}$ and $\mathbf{U}$ are triangular matrices.

Note that only the first k components of $\mathbf{b}_k$ are needed for these computations. Therefore, for each k, we may replace $\mathbf{b}_k$ by its projection $\overline{\mathbf{b}}_k$ onto the subspace of $\mathbf{R}^{m\times 1}$ generated by the first k unit vectors.

Example:

Consider the 3×3 system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix}$.

We know from Section **1.4** that the factorization $\mathbf{A} = \mathbf{E}_3\mathbf{E}_2\mathbf{A}_1$ entails the following auxiliary quantities:

$$\mathbf{u}_1^T = (\,1 \quad * \quad *\,) \qquad A_{11}^{-1} = \tfrac{1}{8}$$

$$\omega_2 = (\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,) \qquad \theta_2^{-1} = \tfrac{40}{37}$$

$$\omega_3 = (\,\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}\,) \qquad \theta_3^{-1} = -\tfrac{37}{180}$$

A straightforward application of Algorithm **2.2** yields

$$\overline{\mathbf{b}}_3 = \begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix} \qquad x_3 = -\tfrac{37}{180}\,(\,\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}\,)\begin{pmatrix} 24 \\ 14 \\ -8 \end{pmatrix} = 3$$

$$\overline{\mathbf{b}}_2 = \begin{pmatrix} 24 \\ 14 \\ * \end{pmatrix} - 3\begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix} = \begin{pmatrix} 6 \\ -7 \\ * \end{pmatrix} \qquad x_2 = \tfrac{40}{37}\,(\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,)\begin{pmatrix} 6 \\ -7 \\ * \end{pmatrix} = -2$$

$$\overline{\mathbf{b}}_1 = \begin{pmatrix} 6 \\ * \\ * \end{pmatrix} + 2\begin{pmatrix} 1 \\ * \\ * \end{pmatrix} = \begin{pmatrix} 8 \\ * \\ * \end{pmatrix} \qquad x_1 = \tfrac{1}{8}\,(\,1 \quad * \quad *\,)\begin{pmatrix} 8 \\ * \\ * \end{pmatrix} = 1$$

The final result is the same as in Section **1.5**:

$$\mathbf{x} = \begin{pmatrix} 1 \\ -2 \\ 3 \end{pmatrix}$$

2.3 Solution of $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$

As in Section **2.2**, we can streamline the method explained in Section **1.6** and avoid solving the complete triangular system involving $\mathbf{A}_1$. For $1 \leq k \leq m$, define $\overline{\boldsymbol{\gamma}}_k$ as the projection of $\boldsymbol{\gamma}$ onto the subspace $\mathbf{S}_k$ of $\mathbf{R}^{1\times m}$ generated by the first k unit vectors. Then define $\overline{\boldsymbol{\pi}}_k$ as the unique solution of the system $\overline{\boldsymbol{\pi}}_k \mathbf{A}_k = \overline{\boldsymbol{\gamma}}_k$. Now, consider the following Lemmas:

Lemma 2.3.1 For $1 \leq k \leq m$, $\overline{\boldsymbol{\pi}}_k$ is in $\mathbf{S}_k$.

Proof The last $m - k$ equations of $\overline{\boldsymbol{\pi}}_k \mathbf{A}_k = \overline{\boldsymbol{\gamma}}_k$ constitute a full rank triangular system with null right hand side. Its solution, a vector made of the last $m - k$ components of $\overline{\boldsymbol{\pi}}_k$, must be zero.

Lemma 2.3.2 For $2 \leq k \leq m$, $(\overline{\boldsymbol{\pi}}_k - \overline{\boldsymbol{\pi}}_{k-1})\,\mathbf{A}_k = (\gamma_k - \overline{\boldsymbol{\pi}}_{k-1}\mathbf{v}_k)\,\mathbf{u}_k^T$, where γ_k is the k^{th} component of $\boldsymbol{\gamma}$ or, equivalently, of $\overline{\boldsymbol{\gamma}}_k$.

Proof Using essentially the definitions of $\overline{\boldsymbol{\gamma}}_k$ and $\overline{\boldsymbol{\gamma}}_{k-1}$, we obtain

$$\begin{aligned}
\overline{\boldsymbol{\pi}}_k \mathbf{A}_k &= \overline{\boldsymbol{\gamma}}_k \\
&= \overline{\boldsymbol{\gamma}}_{k-1} + \gamma_k \mathbf{u}_k^T \\
&= \overline{\boldsymbol{\pi}}_{k-1}\mathbf{A}_{k-1} + \gamma_k \mathbf{u}_k^T \\
&= \overline{\boldsymbol{\pi}}_{k-1}\,(\mathbf{A}_k - \mathbf{v}_k\mathbf{u}_k^T) + \gamma_k \mathbf{u}_k^T \\
&= \overline{\boldsymbol{\pi}}_{k-1}\mathbf{A}_k + (\gamma_k - \overline{\boldsymbol{\pi}}_{k-1}\mathbf{v}_k)\,\mathbf{u}_k^T.
\end{aligned}$$

Lemma 2.3.3 For $2 \leq k \leq m$, $\overline{\boldsymbol{\pi}}_k = \overline{\boldsymbol{\pi}}_{k-1} + \theta_k^{-1}\,(\gamma_k - \overline{\boldsymbol{\pi}}_{k-1}\mathbf{v}_k)\,\boldsymbol{\omega}_k$.

Proof The matrix $\mathbf{A}_k$ is non-singular and we have

$$\begin{aligned}
(\overline{\boldsymbol{\pi}}_k - \overline{\boldsymbol{\pi}}_{k-1})\,\mathbf{A}_k &= (\gamma_k - \overline{\boldsymbol{\pi}}_{k-1}\mathbf{v}_k)\,\mathbf{u}_k^T \\
&= (\gamma_k - \overline{\boldsymbol{\pi}}_{k-1}\mathbf{v}_k)\,\theta_k^{-1}\boldsymbol{\omega}_k\mathbf{A}_k\ .
\end{aligned}$$

By Lemma **2.3.1**, only the first component of $\overline{\boldsymbol{\pi}}_1$ can be nonzero. Therefore, the system $\overline{\boldsymbol{\pi}}_1\mathbf{A}_1 = \overline{\boldsymbol{\gamma}}_1$ can be solved in one scalar division as opposed to the system $\boldsymbol{\pi}_1\mathbf{A}_1 = \boldsymbol{\gamma}$ which requires a triangular matrix division. Then, Lemma **2.3.3** shows that the sequence $\overline{\boldsymbol{\pi}}_k$ $(2 \leq k \leq m)$ can be computed just as easily as the sequence $\boldsymbol{\pi}_k$ $(2 \leq k \leq m)$ of Section **1.5**. Finally, since $\boldsymbol{\gamma} = \overline{\boldsymbol{\gamma}}_m$, we have $\boldsymbol{\pi} = \overline{\boldsymbol{\pi}}_m$. In summary, we obtain the following method.

Algorithm 2.3 (to solve $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$)

Let $\overline{\boldsymbol{\pi}}_1 = \gamma_1 A_{11}^{-1} \mathbf{u}_1^T$.

For $k = 2$ to m , let $\xi_k = \gamma_k - \overline{\boldsymbol{\pi}}_{k-1}\mathbf{v}_k$ and $\overline{\boldsymbol{\pi}}_k = \overline{\boldsymbol{\pi}}_{k-1} + \xi_k \theta_k^{-1} \omega_k$.

Let $\boldsymbol{\pi} = \overline{\boldsymbol{\pi}}_m$.

Once again, this algorithm actually depends on the vectors $\mathbf{u}_k^T\mathbf{A}_k^{-1}$ $(1 \le k \le m)$. The maximum number of multiplications required is

$$1 + \sum_{k=2}^{m} 2k \approx m^2 + m.$$

Example:

Consider the 3×3 system $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ where $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$ and $\boldsymbol{\gamma} = (6 \quad 0 \quad -6)$. We know from Section **1.4** that the factorization $\mathbf{A} = \mathbf{E}_3\mathbf{E}_2\mathbf{A}_1$ entails the following auxiliary quantities:

$$\mathbf{u}_1^T = (1 \quad * \quad *) \qquad A_{11}^{-1} = \tfrac{1}{8}$$

$$\omega_2 = (-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *) \qquad \theta_2^{-1} = \tfrac{40}{37}$$

$$\omega_3 = (\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}) \qquad \theta_3^{-1} = -\tfrac{37}{180}$$

A straightforward application of Algorithm **2.3** yields

$$\overline{\boldsymbol{\pi}}_1 = 6\,\tfrac{1}{8}\,(1 \quad * \quad *) = (\tfrac{3}{4} \quad * \quad *)$$

$$\xi_2 = 0 - (\tfrac{3}{4} \quad * \quad *)\begin{pmatrix} 1 \\ * \\ * \end{pmatrix} = -\tfrac{3}{4}$$

$$\overline{\boldsymbol{\pi}}_2 = (\tfrac{3}{4} \quad * \quad *) - \tfrac{3}{4}\,\tfrac{40}{37}\,(-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *) = (\tfrac{30}{37} \quad -\tfrac{6}{37} \quad *)$$

$$\xi_3 = -6 - (\tfrac{30}{37} \quad -\tfrac{6}{37} \quad *)\begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix} = -\tfrac{360}{37}$$

$$\overline{\boldsymbol{\pi}}_3 = (\tfrac{30}{37} \quad -\tfrac{6}{37} \quad *) + \tfrac{360}{37}\,\tfrac{37}{180}\,(\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}) = (1 \quad -2 \quad 1).$$

The final result is the same as in Section **1.6**:

$$\boldsymbol{\pi} = (1 \quad -2 \quad 1).$$

2.4 Computation of the Factors $\mathbf{E}_k$ $(2 \le k \le m)$

The system $\boldsymbol{\omega}_k \mathbf{A}_{k-1} = \mathbf{u}_k^T$ defining $\boldsymbol{\omega}_k$ can be written

$$\boldsymbol{\omega}_k \begin{pmatrix} \overline{\mathbf{A}} & 0 & 0 \\ \boldsymbol{\rho} & \alpha & 0 \\ \mathbf{B} & \mathbf{c} & \mathbf{D} \end{pmatrix} = (\,0 \quad 1 \quad 0\,)$$

where $\overline{\mathbf{A}}$ is the $(k-1) \times (k-1)$ leading submatrix of $\mathbf{A}$ or $\mathbf{A}_{k-1}$, and α is the k^{th} diagonal element of $\mathbf{A}$ or $\mathbf{A}_{k-1}$. Assuming that $\mathbf{A}_{k-1}$ is non-singular, this system is equivalent to

$$\boldsymbol{\omega}_k \;=\; \alpha^{-1}(\,\boldsymbol{\pi} \quad 1 \quad 0\,) \qquad \text{and} \qquad \boldsymbol{\pi}\overline{\mathbf{A}} = -\boldsymbol{\rho}.$$

From the relationship $\mathbf{A}_{k-1} = (\mathbf{I} + \mathbf{v}_{k-1}\boldsymbol{\omega}_{k-1})(\mathbf{I} + \mathbf{v}_{k-2}\boldsymbol{\omega}_{k-2}) \ldots (\mathbf{I} + \mathbf{v}_2\boldsymbol{\omega}_2)$ $\mathbf{A}_1$, we derive $\overline{\mathbf{A}} = (\mathbf{I} + \overline{\mathbf{v}}_{k-1}\overline{\boldsymbol{\omega}}_{k-1})(\mathbf{I} + \overline{\mathbf{v}}_{k-2}\overline{\boldsymbol{\omega}}_{k-2}) \ldots (\mathbf{I} + \overline{\mathbf{v}}_2\overline{\boldsymbol{\omega}}_2)\overline{\mathbf{A}}_1$ where $\overline{\mathbf{v}}_l$ is the $(k-1) \times 1$ leading submatrix of $\mathbf{v}_l$, $\overline{\boldsymbol{\omega}}_l$ is the $1 \times (k-1)$ leading submatrix of $\boldsymbol{\omega}_l$ and $\overline{\mathbf{A}}_1$ is the $(k-1) \times (k-1)$ leading submatrix of $\mathbf{A}_1$.

When we compute the vector $\boldsymbol{\omega}_k$, we already know the vectors $\boldsymbol{\omega}_l$ $(2 \le l \le k-1)$ and hence the vectors $\overline{\boldsymbol{\omega}}_l$ $(2 \le l \le k-1)$. Therefore, we can apply Algorithm **2.3** to solve the $(k-1)$–dimensional system $\boldsymbol{\pi}\overline{\mathbf{A}} = -\boldsymbol{\rho}$.

The maximum number of multiplications or divisions required to compute $(\boldsymbol{\omega}_k, \theta_k)$ by this method is

$$k^2 - k \quad + \quad k \quad + \quad k - 1 \;\approx\; k^2 + k$$

and the maximum number of multiplications required to factorize $\mathbf{A}$ is

$$\sum_{k=2}^{m} k^2 + k \;\approx\; \tfrac{1}{3}m^3 + m^2.$$

This is essentially the same as the maximum number of multiplications required to compute the triangular factorization $\mathbf{A} = \mathbf{LU}$ by Gaussian elimination.

Example:

Consider the 3×3 matrix $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$.

- We have $A_{11} = 8$.
- The factor $\mathbf{E}_2 = \mathbf{I} + \mathbf{v}_2 \boldsymbol{\omega}_2$ is computed from $\boldsymbol{\omega}_2 \mathbf{A}_1 = \mathbf{u}_2^T$.

$$\alpha = 5$$

$$\boldsymbol{\pi}(\,8\,) = (\,-3\,)$$

$$\boldsymbol{\pi} = (\,-\tfrac{3}{8}\,)$$

$$\boldsymbol{\omega}_2 = \tfrac{1}{5}(\,-\tfrac{3}{8} \quad 1 \quad *\,) = (\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,)$$

$$\theta_2 = 1 + (\,-\tfrac{3}{40} \quad \tfrac{1}{5} \quad *\,)\begin{pmatrix} 1 \\ * \\ * \end{pmatrix} = \tfrac{37}{40}$$

- The factor $\mathbf{E}_3 = \mathbf{I} + \mathbf{v}_3 \boldsymbol{\omega}_3$ is computed from $\boldsymbol{\omega}_3 \mathbf{A}_2 = \mathbf{u}_3^T$.

$$\alpha = 2$$

$$\boldsymbol{\pi}\begin{pmatrix} 8 & 3 \\ 1 & 5 \end{pmatrix} = (\,-4 \quad -9\,)$$

$$\bar{\boldsymbol{\pi}}_1 = (\,-\tfrac{1}{2} \quad *\,)$$

$$\bar{\boldsymbol{\pi}}_2 = (\,-\tfrac{1}{2} \quad *\,) + \left[-9 - (\,-\tfrac{1}{2} \quad *\,)\begin{pmatrix} 1 \\ * \end{pmatrix}\right] \tfrac{40}{37}(\,-\tfrac{3}{40} \quad \tfrac{1}{5}\,)$$

$$\boldsymbol{\pi} = (\,\tfrac{7}{37} \quad -\tfrac{68}{37}\,)$$

$$\boldsymbol{\omega}_3 = \tfrac{1}{2}(\,\tfrac{7}{37} \quad -\tfrac{68}{37} \quad 1\,) = (\,\tfrac{7}{74} \quad -\tfrac{34}{37} \quad \tfrac{1}{2}\,)$$

$$\theta_3 = 1 + (\,\tfrac{7}{74} - \tfrac{34}{37} \quad \tfrac{1}{2}\,)\begin{pmatrix} 6 \\ 7 \\ * \end{pmatrix} = -\tfrac{180}{37}$$

2.5 Simplification Applicable to Non-Spikes

We saw in Section **1.7** that if column l is not a spike of **A**, the Basic Method does not require the computation of the auxiliary vector $\boldsymbol{\omega}_l$. This simplification is not directly transferable in the Streamlined Method, because algorithms **2.2** and **2.3** appear to require the computation of all the intermediate quantities like x_l or $\bar{\boldsymbol{\pi}}_l$, and hence of all the vectors $\boldsymbol{\omega}_l$.

However, if the spikes of $\mathbf{A}$ were its rightmost columns, the leading $t \times t$ submatrix $\mathbf{T}$ associated with the t non-spike columns of $\mathbf{A}$ would be triangular. Then, the vectors $\theta_l^{-1}\boldsymbol{\omega}_l$ $(1 \leq l \leq t)$ would represent the rows of $\mathbf{T}^{-1}$. Moreover, the steps of algorithms **2.2** and **2.3** corresponding to spike columns k could be carried out as earlier with the vectors $\boldsymbol{\omega}_k$, while those corresponding to nonspike columns l could be replaced by solving a system explicitly involving the triangular matrix $\mathbf{T}$ instead of the vectors $\boldsymbol{\omega}_l$.

In order to implement this idea when the matrix $\mathbf{A}$ is arbitrary, we can permute the columns of $\mathbf{A}$ according to the permutation matrix $\mathbf{Q}$ that sends the spikes of $\mathbf{A}$ to the right while preserving the order of the non-spikes and that of the spikes, and permute the rows of $\mathbf{A}$ symmetrically, i.e. according to the permutation matrix $\mathbf{Q}^T$. Let $\mathbf{A}' \triangleq \mathbf{Q}^T\mathbf{A}\mathbf{Q}$ be the resulting matrix. Let $\mathbf{q}$ be the permutation of $\{1, 2, \ldots, m\}$ induced by $\mathbf{Q}$ (i.e. $\mathbf{u}_{\mathbf{q}(k)} = \mathbf{Q}^T\mathbf{u}_k$). Then we have the following Lemma:

Lemma 2.5.1 Column j is a spike of $\mathbf{A}$ if and only if column $\mathbf{q}(j)$ is a spike of $\mathbf{A}'$.

Proof If column j is a spike, then $\exists i \quad i < j$ and $A_{ij} \neq 0$. By construction, if column j is a spike of $\mathbf{A}$, then $i < j \implies \mathbf{q}(i) < \mathbf{q}(j)$. In addition, $A'_{\mathbf{q}(i)\mathbf{q}(j)} = \mathbf{u}^T_{\mathbf{q}(i)}\mathbf{A}'\mathbf{u}_{\mathbf{q}(j)} = \mathbf{u}_i^T\mathbf{Q}\ \mathbf{Q}^T\mathbf{A}\mathbf{Q}\ \mathbf{Q}^T\mathbf{u}_j = \mathbf{u}_i^T\mathbf{A}\mathbf{u}_j = A_{ij} \neq 0$. Therefore, column $\mathbf{q}(j)$ is a spike of $\mathbf{A}'$.

Conversely, if column $\mathbf{q}(j)$ is a spike of $\mathbf{A}'$, then $\exists i\ \mathbf{q}(i) < \mathbf{q}(j)andA'_{\mathbf{q}(i)\mathbf{q}(j)} \neq 0$. Either $i < j$ or $j < i$. In the former case, $\exists i \quad i < j$ and $A_{ij} \neq 0$. In the latter case, column j which stood to the left of column i in $\mathbf{A}$ will stand to the [illegible] column i in $\mathbf{A}'$. Either way, column j must be a spike of $\mathbf{A}$.

By Lemma **2.5.1**, $\mathbf{A}$ and $\mathbf{A}'$ have the same number s of spikes and the same number $t = m - s$ of non-spikes. Since the t leftmost columns of $\mathbf{A}'$ are not spikes, the decomposition

$$\mathbf{A}' = \mathbf{A}'_1 + \mathbf{v}'_2\mathbf{u}_2^T + \mathbf{v}'_3\mathbf{u}_3^T + \cdots + \mathbf{v}'_m\mathbf{u}_m^T = \mathbf{E}'_m\ \mathbf{E}'_{m-1} \ldots \mathbf{E}'_2\ \mathbf{A}'_1$$

satisfies $\mathbf{v}'_2 = \mathbf{v}'_3 = \cdots = \mathbf{v}'_t = \mathbf{0}$ and $\mathbf{E}'_2 = \mathbf{E}'_3 = \cdots = \mathbf{E}'_t = \mathbf{I}$. In particular, $\mathbf{A}'_t$ is triangular.

To solve $\mathbf{Ax} = \mathbf{b}$, we consider the equivalent system $\mathbf{A}'\mathbf{x}' = \mathbf{b}'$ where $\mathbf{x}' \triangleq \mathbf{Q}^T\mathbf{x}$ and $\mathbf{b}' \triangleq \mathbf{Q}^T\mathbf{b}$. The last s components of $\mathbf{x}'$ are computed as in Section **2.2** using the vectors $\boldsymbol{\omega}'_k$ $(m \geq k > t)$, and the first t components of $\mathbf{x}'$ are solved through the first t equations of the triangular system $\mathbf{A}'_t\mathbf{x}' = \mathbf{b}'_t$.

To solve $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$, we consider the equivalent system $\boldsymbol{\pi}'\mathbf{A}' = \boldsymbol{\gamma}'$ where $\boldsymbol{\pi}' \triangleq \boldsymbol{\pi}\mathbf{Q}$ and $\boldsymbol{\gamma}' \triangleq \boldsymbol{\gamma}\mathbf{Q}$. The vector $\overline{\boldsymbol{\pi}}'_t$ is given by the triangular system $\overline{\boldsymbol{\pi}}'_t\mathbf{A}'_t = \overline{\boldsymbol{\gamma}}'_t$ and the last s vectors $\overline{\boldsymbol{\pi}}'_k$ are computed as in Section **2.3** using the vectors $\boldsymbol{\omega}'_k$ $(t < k \leq m)$.

To compute the non-trivial factors $\mathbf{E}'_k$ $(t < k \leq m)$, we decompose the system $\boldsymbol{\omega}'_k\mathbf{A}'_{k-1} = \mathbf{u}_k^T$ into $\boldsymbol{\omega}'_k = \alpha^{-1}(\boldsymbol{\pi} \quad 1 \quad \mathbf{0})$ and $\boldsymbol{\pi}\overline{\mathbf{A}} = -\boldsymbol{\rho}$ as shown in Section **2.4** and solve the latter system as indicated in the previous paragraph.

That way, all computations can be carried out solely with the auxiliary quantities $(\boldsymbol{\omega}'_k, \theta'_k)$ $(t < k \leq m)$ corresponding to spikes $(\mathbf{v}'_k \neq \mathbf{0})$. Because the spikes have been shifted to the right in $\mathbf{A}'$, the number of nonzeros above the diagonal in the spikes may increase. However, the number of nonzeros in the vectors $\boldsymbol{\omega}'_k$ remains unchanged as the following Lemma indicates:

Lemma 2.5.2 If column k is a spike of $\mathbf{A}$, then $\boldsymbol{\omega}'_{\mathrm{q}(k)} = \boldsymbol{\omega}_k\mathbf{Q}$ and $\theta'_{\mathrm{q}(k)} = \theta_k$.

Proof Let $k' = \mathrm{q}(k)$. We have, by definition of $\boldsymbol{\omega}'_{k'}$,

$$\boldsymbol{\omega}'_{k'}\mathbf{A}'_{k'-1} = \mathbf{u}_{k'}^T$$

and, by definition of $\boldsymbol{\omega}_k$,

$$\boldsymbol{\omega}_k\mathbf{A}_{k-1} = \mathbf{u}_k^T$$

$$\boldsymbol{\omega}_k\mathbf{Q}\ \mathbf{Q}^T\mathbf{A}_{k-1}\mathbf{Q} = \mathbf{u}_k^T\mathbf{Q} = \mathbf{u}_{\mathrm{q}(k)}^T = \mathbf{u}_{k'}^T\ .$$

Partition $\mathbf{Q}^T\mathbf{A}_{k-1}\mathbf{Q}$ into $\begin{pmatrix} \mathbf{X} & \mathbf{Z} \\ \mathbf{Y} & \mathbf{T} \end{pmatrix}$ where $\mathbf{X}$ is a $k' \times k'$ submatrix. Then $\mathbf{Z} = \mathbf{0}$, $\mathbf{X}$ and $\mathbf{T}$ have full rank and $\mathbf{X}$ equals the leading $k' \times k'$ submatrix of

$\mathbf{A}' = \mathbf{Q}^T\mathbf{A}\mathbf{Q}$. Therefore, the last $m - k'$ components of $\boldsymbol{\omega}'_{k'}$ and of $\boldsymbol{\omega}_k\mathbf{Q}$ are zeros while their first k' components are solutions of the same well determined system. Thus $\boldsymbol{\omega}'_{\mathbf{q}(k)} = \boldsymbol{\omega}_k\mathbf{Q}$.

Finally, since the first k' components of $\mathbf{v}'_{\mathbf{q}(k)}$ and $\mathbf{Q}_k^T\mathbf{v}_k$ are equal, we have

$$\begin{aligned}\theta'_{\mathbf{q}(k)} &= 1 + \boldsymbol{\omega}'_{\mathbf{q}(k)}\mathbf{v}'_{\mathbf{q}(k)} \\ &= 1 + \boldsymbol{\omega}_k\mathbf{Q}_k\mathbf{Q}_k^T\mathbf{v}_k \\ &= 1 + \boldsymbol{\omega}_k\mathbf{v}_k \\ &= \theta_k \ .\end{aligned}$$

Example:

This example shows how to permute and factorize a 5 × 5 matrix whose columns 2 and 4 are not spikes, and how to solve systems like $\mathbf{A}\mathbf{x} = \mathbf{b}$ or $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$. In this case, the permutation $\mathbf{Q}$ will simply interchange columns 3 and 4.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} 6 \\ 1 \\ 6 \\ 6 \\ 8 \end{pmatrix} \qquad \boldsymbol{\gamma} = (1 \quad 3 \quad 3 \quad 2 \quad 3)$$

$$\mathbf{A}' = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad \mathbf{b}' = \begin{pmatrix} 6 \\ 1 \\ 6 \\ 6 \\ 8 \end{pmatrix} \qquad \boldsymbol{\gamma}' = (1 \quad 3 \quad 2 \quad 3 \quad 3)$$

$$\mathbf{A}'_1 = \mathbf{A}'_2 = \mathbf{A}'_3 = \begin{pmatrix} 1 & * & * & * & * \\ 0 & 1 & * & * & * \\ 0 & 1 & 1 & * & * \\ -1 & 0 & * & 1 & * \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad \begin{array}{lcl} \mathbf{v}'_2 = \mathbf{0} & \Longrightarrow & \mathbf{E}'_2 = \mathbf{I} \\ \mathbf{v}'_3 = \mathbf{0} & \Longrightarrow & \mathbf{E}'_3 = \mathbf{I} \end{array}$$

$$\mathbf{A}'_4 = \begin{pmatrix} 1 & * & * & 1 & * \\ 0 & 1 & * & 0 & * \\ 0 & 1 & 1 & 1 & * \\ -1 & 0 & * & 1 & * \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad \mathbf{v}'_4 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ * \\ * \end{pmatrix}$$

$$\mathbf{A}'_5 = \begin{pmatrix} 1 & * & * & 1 & 1 \\ 0 & 1 & * & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ -1 & 0 & * & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad \mathbf{v}'_5 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ * \end{pmatrix}$$

- The factor $\mathbf{E}'_4 \;=\; \mathbf{I} + \mathbf{v}'_4 \boldsymbol{\omega}'_4$ is computed from $\boldsymbol{\omega}'_4 \mathbf{A}'_3 \;=\; \mathbf{u}_4^T$.

$$\alpha \;=\; 1$$

$$\boldsymbol{\pi} \begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix} \;=\; (\,1 \quad 0\,)$$

$$\boldsymbol{\pi} \;=\; (\,1 \quad 0\,)$$

$$\boldsymbol{\omega}'_4 \;=\; 1^{-1}(\,1 \quad 0 \quad * \quad 1 \quad *\,) \;=\; (\,1 \quad 0 \quad * \quad 1 \quad *\,)$$

$$\theta'_4 \;=\; 1 \;+\; (\,1 \quad 0 \quad * \quad 1 \quad *\,) \begin{pmatrix} 1 \\ 0 \\ 1 \\ * \\ * \end{pmatrix} \;=\; 2$$

- The factor $\mathbf{E}'_5 \;=\; \mathbf{I} + \mathbf{v}'_5 \boldsymbol{\omega}'_5$ is computed from $\boldsymbol{\omega}'_5 \mathbf{A}'_4 \;=\; \mathbf{u}_5^T$.

$$\alpha \;=\; 1$$

$$\boldsymbol{\pi} \begin{pmatrix} 1 & * & * & 1 \\ 0 & 1 & * & 0 \\ 0 & 1 & 1 & 1 \\ -1 & 0 & * & 1 \end{pmatrix} \;=\; (\,-1 \quad -1 \quad -1 \quad 0\,)$$

$$\bar{\boldsymbol{\pi}}_3 \;=\; (\,-1 \quad 0 \quad -1 \quad *\,)$$

$$\overline{\boldsymbol{\pi}}_4 = (\,-1 \quad 0 \quad -1 \quad *\,) + \left[0 - (\,-1 \quad 0 \quad -1 \quad *\,) \begin{pmatrix} 1 \\ 0 \\ 1 \\ * \end{pmatrix} \right] \tfrac{1}{2}(\,1 \quad 0 \quad * \quad 1\,)$$

$$\boldsymbol{\pi} = (\,0 \quad 0 \quad -1 \quad 1\,)$$

$$\boldsymbol{\omega}_5' = 1^{-1}(\,0 \quad 0 \quad -1 \quad 1 \quad 1\,) = (\,0 \quad 0 \quad -1 \quad 1 \quad 1\,)$$

$$\theta_5' = 1 + (\,0 \quad 0 \quad -1 \quad 1 \quad 1\,) \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ * \end{pmatrix} = 2$$

- The vector $\mathbf{x}$ is computed from $\mathbf{E}_5'\mathbf{E}_4'\mathbf{A}_3'\mathbf{x}' = \mathbf{b}'$.

$$\overline{\mathbf{b}}_5' = \begin{pmatrix} 6 \\ 1 \\ 6 \\ 6 \\ 8 \end{pmatrix} \qquad x_5' = \tfrac{1}{2}(\,0 \quad 0 \quad -1 \quad 1 \quad 1\,) \begin{pmatrix} 6 \\ 1 \\ 6 \\ 6 \\ 8 \end{pmatrix} = 4$$

$$\overline{\mathbf{b}}_4' = \begin{pmatrix} 6 \\ 1 \\ 6 \\ 6 \\ * \end{pmatrix} - 4 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ * \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 6 \\ 2 \\ * \end{pmatrix} \qquad x_4' = \tfrac{1}{2}(\,1 \quad 0 \quad 0 \quad 1 \quad *\,) \begin{pmatrix} 2 \\ 1 \\ 2 \\ 6 \\ 8 \end{pmatrix} = 2$$

$$\overline{\mathbf{b}}_3' = \begin{pmatrix} 2 \\ 1 \\ 6 \\ * \\ * \end{pmatrix} - 2 \begin{pmatrix} 1 \\ 0 \\ 1 \\ * \\ * \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 4 \\ * \\ * \end{pmatrix} \qquad \begin{pmatrix} 1 & * & * \\ 0 & 1 & * \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$$

$$\mathbf{x}' = \begin{pmatrix} 0 \\ 1 \\ 3 \\ 2 \\ 4 \end{pmatrix} \qquad \mathbf{x} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

- The vector $\boldsymbol{\pi}$ is computed from $\boldsymbol{\pi}'\mathbf{E}_5'\mathbf{E}_4'\mathbf{A}_3' = \boldsymbol{\gamma}'$.

$$\overline{\boldsymbol{\pi}}_3'\mathbf{A}_3' = (\,1 \quad 3 \quad 2 \quad * \quad *\,)$$

$$\overline{\boldsymbol{\pi}}_3' = (\,1 \quad 1 \quad 2 \quad * \quad *\,)$$

$$\begin{aligned}
\overline{\boldsymbol{\pi}}'_4 &= (1 \quad 1 \quad 2 \quad * \quad *) + \left[3 - (1 \quad 1 \quad 2 \quad * \quad *) \begin{pmatrix} 1 \\ 0 \\ 1 \\ * \\ * \end{pmatrix} \right] \tfrac{1}{2} (1 \quad 0 \quad * \quad 1 \quad *) \\
&= (1 \quad 1 \quad 2 \quad 0 \quad *) \\
\overline{\boldsymbol{\pi}}'_5 &= (1 \quad 1 \quad 2 \quad 0 \quad *) + \left[3 - (1 \quad 1 \quad 2 \quad 0 \quad *) \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ * \end{pmatrix} \right] \tfrac{1}{2} (0 \quad 0 \quad -1 \quad 1 \quad 1) \\
&= (1 \quad 1 \quad 1 \quad 1 \quad 1)
\end{aligned}$$

$$\boldsymbol{\pi}' = (1 \quad 1 \quad 1 \quad 1 \quad 1)$$

$$\boldsymbol{\pi} = (1 \quad 1 \quad 1 \quad 1 \quad 1)$$

CHAPTER 3. NUMERICAL STABILITY

3.1 Introduction

In Chapter 2, we showed how to factorize a matrix $\mathbf{A}$ under the assumption that the matrices $\mathbf{A}_k$ $(1 \leq k \leq m)$ were non-singular. In this Chapter, we decompose this assumption into two conditions and show that a rescaled version of the same factorization can be obtained under only the first condition. We recall that if $\mathbf{A}$ is non-singular then its columns can be permuted so that the resulting matrix satisfies the first condition, and indicate a procedure to carry out simultaneously the permutation and the factorization. This procedure offers a numerical stability similar to that of LU decomposition with partial pivoting.

3.2 Definitions and Notation

A unit upper triangular matrix is an upper triangular matrix whose entries on the diagonal are equal to one.

Let $\mathbf{A}$ be a square matrix. Let $\mathbf{B}$ be a square submatrix of $\mathbf{A}$. We define the minor of $\mathbf{A}$ associated with $\mathbf{B}$ as the determinant of $\mathbf{B}$. A leading minor of $\mathbf{A}$ is a minor associated with a leading submatrix of $\mathbf{A}$.

Let $\mathbf{A}$,$\mathbf{v}$ and $\boldsymbol{\omega}$ be respectively an $m \times n$, an $m \times 1$ and a $1 \times n$ matrix. Then, for $k \leq m$ and $k \leq n$, $\overline{(\mathbf{A})_k}$, $\overline{(\mathbf{v})_k}$ and $\overline{(\boldsymbol{\omega})_k}$ denote respectively the $k \times k$ leading submatrix of $\mathbf{A}$, the $k \times 1$ leading submatrix of $\mathbf{v}$ and the $1 \times k$ leading submatrix of $\boldsymbol{\omega}$. Unless indicated otherwise, $\overline{\mathbf{A}}_k$, $\overline{\mathbf{v}}_k$ and $\overline{\boldsymbol{\omega}}_k$ are abbreviations for $\overline{(\mathbf{A}_k)_k}$, $\overline{(\mathbf{v}_k)_k}$ and $\overline{(\boldsymbol{\omega}_k)_k}$.

3.3 Conditions on the Matrices $\mathbf{A}_k$ $(1 \leq k \leq m)$

In Chapter 2, we assumed that the matrices $\mathbf{A}_k$ $(1 \leq k \leq m)$ were non-singular. Because of their structure, we have

$$\det \mathbf{A}_k \;=\; \det \overline{\mathbf{A}}_k \prod_{l=k+1}^{m} A_{ll} \;=\; \det \overline{\mathbf{A}}_k \prod_{l=k+1}^{m} (A_1)_{ll}.$$

Therefore, the matrices $\mathbf{A}_k$ $(1 \leq k \leq m)$ are non-singular if and only if the matrices $\overline{\mathbf{A}}_k$ $(1 \leq k \leq m)$ and the matrix $\mathbf{A}_1$ are non-singular.

It is well known that if $\mathbf{A}$ is non-singular, then its columns can be permuted according to a permutation matrix $\mathbf{Q}$ such that the leading square submatrices of $\mathbf{AQ}$ are non-singular. Actually, the columns of $\mathbf{A}$ can be selected step by step: at step k, a column is chosen for k^{th} position so as to maximize the absolute value of the resulting k^{th} leading minor. In practice, this procedure yields well conditioned leading submatrices, as observed in the LU decomposition algorithm with *column* interchanges. Therefore, we shall follow a similar procedure.

Regarding the non-singularity of $(\mathbf{AQ})_1$, i.e. the absence of zeros on the diagonal of $\mathbf{AQ}$, we shall see in Section **3.6** that the issue is rendered moot by rescaling the vectors representing the factorization.

3.4 LU Decomposition with Column Interchanges

In this Section, we recall some useful properties of the LU decomposition of square matrices (Murty, 1976).

Lemma 3.4.1 For any $m \times m$ matrix $\mathbf{A}$, there exist a lower triangular matrix $\mathbf{L}$, $m-1$ transposition matrices $\mathbf{T}_k$ $(1 \leq k < m)$ and $m-1$ unit upper triangular matrices $\mathbf{U}_k$ $(1 \leq k < m)$ such that

$$\mathbf{L} \;=\; \mathbf{AT}_1\mathbf{U}_1\mathbf{T}_2\mathbf{U}_2 \ldots \mathbf{T}_{m-1}\mathbf{U}_{m-1}.$$

Proof (and algorithm 3.4) Let $\mathbf{A}^{(0)} = \mathbf{A}$. Given $\mathbf{A}^{(k-1)}$, let j_k satisfy $|A^{(k-1)}_{k,j_k}| \;=\; \max_{k \leq j \leq m} |A^{(k-1)}_{k,j}|$.

If $A^{(k-1)}_{k,j_k} = 0$, $\mathbf{A}^{(k-1)}$ is singular. Let $\mathbf{T}_k = \mathbf{U}_k = \mathbf{I}$.

Otherwise, interchange columns k and j_k, i.e. postmultiply $\mathbf{A}^{(k-1)}$ by the transposition matrix $\mathbf{T}_{k,j_k}$ (abbreviated to $\mathbf{T}_k$) derived from the identity matrix by interchanging columns k and j_k. Then zero out the entries of row k to the right of column k by adding appropriate multiples of column k to columns $k+1, \ldots, m$, i.e. postmultiply $\mathbf{A}^{(k-1)}\mathbf{T}_{k,j_k}$ by a unit upper triangular matrix $\mathbf{U}_k$ with the appropriate entries in row k.

In either case, we have

$$\mathbf{A}^{(k)} = \mathbf{A}^{(k-1)}\mathbf{T}_k\mathbf{U}_k \qquad \text{for } 1 \le k \le m-1$$

where the strictly upper triangular part of $\mathbf{A}^{(k)}$ has zero entries in its k first rows. After $m-1$ iterations, we obtain $\mathbf{A}^{(m-1)} = \mathbf{A}^{(0)}\mathbf{T}_1\mathbf{U}_1\mathbf{T}_2\mathbf{U}_2\ldots\mathbf{T}_{m-1}\mathbf{U}_{m-1}$ where $\mathbf{A}^{(m-1)}$ is lower triangular.

Lemma 3.4.2 For $0 \le k \le m-1$, let $\mathbf{A}^{(k)} = \mathbf{A}\mathbf{T}_1\mathbf{U}_1\mathbf{T}_2\mathbf{U}_2\ldots\mathbf{T}_k\mathbf{U}_k$ be the k^{th} iterate in the LU decomposition of $\mathbf{A}$. Then $\mathbf{A}^{(k)} = \mathbf{A}\mathbf{T}_1\mathbf{T}_2\ldots\mathbf{T}_k\mathbf{U}'_k$ where $\mathbf{U}'_k$ is a unit upper triangular matrix whose last $m-k$ rows equal those of the identity matrix.

Proof (by induction) $\mathbf{A}^{(0)} = \mathbf{A}\mathbf{I}$. Assume that

$$\mathbf{A}^{(k-1)} = \mathbf{A}\mathbf{T}_1\mathbf{T}_2\ldots\mathbf{T}_{k-1}\mathbf{U}'_{k-1} \qquad \text{where} \quad \mathbf{U}'_{k-1} = \begin{pmatrix} \mathbf{V}_{k-1} & \mathbf{\Omega}_{k-1} \\ \mathbf{0} & \mathbf{I}_{m-k+1} \end{pmatrix}.$$

Then

$$\begin{aligned}
\mathbf{U}'_{k-1}\mathbf{T}_k &= \begin{pmatrix} \mathbf{V} & \mathbf{\Omega} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{V} & \mathbf{\Omega S} \\ \mathbf{0} & \mathbf{S} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{pmatrix}\begin{pmatrix} \mathbf{V} & \mathbf{\Omega S} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \\
&= \mathbf{T}_k\mathbf{V}'_{k-1}
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{A}^{(k)} &= \mathbf{A}^{(k-1)}\mathbf{T}_k\mathbf{U}_k \\
&= \mathbf{A}\mathbf{T}_1\mathbf{T}_2\ldots\mathbf{T}_{k-1}\mathbf{U}'_{k-1}\mathbf{T}_k\mathbf{U}_k \\
&= \mathbf{A}\mathbf{T}_1\mathbf{T}_2\ldots\mathbf{T}_{k-1}\mathbf{T}_k\mathbf{V}'_{k-1}\mathbf{U}_k \\
&= \mathbf{A}\mathbf{T}_1\mathbf{T}_2\ldots\mathbf{T}_{k-1}\mathbf{T}_k\mathbf{U}'_k
\end{aligned}$$

where $\mathbf{V}'_{k-1}, \mathbf{U}_k$ and $\mathbf{U}'_k = \mathbf{V}'_{k-1}\mathbf{U}_k$ are unit upper triangular matrices whose last $m-k$ rows equal those of the identity matrix.

Corollary 3.4.3 There exists a lower triangular matrix L, a unit upper triangular matrix $\mathbf{U}$ and a permutation matrix $\mathbf{Q}$ such that $\mathbf{LU} = \mathbf{AQ}$.

Corollary 3.4.4 For $1 \le l \le k$, the l^{th} leading minor of $\mathbf{A}^{(k)}$ and that of $\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k$ are equal:

$$\det\overline{(\mathbf{A}^{(k)})_l} \;=\; \det\overline{(\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k)_l}.$$

Corollary 3.4.5 If $\mathbf{A}$ is non-singular, the first k leading minors of the product $\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k$ are nonzero:

$$\begin{aligned}\det\overline{(\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k)_l} &= \det\overline{(\mathbf{A}^{(k)})_l}\\ &= A^{(k)}_{1,1}\,A^{(k)}_{2,2}\,\ldots\,A^{(k)}_{l,l}\\ &= A^{(m-1)}_{1,1}\,A^{(m-1)}_{2,2}\,\ldots\,A^{(m-1)}_{l,l}.\end{aligned}$$

Corollary 3.4.6 If $\mathbf{A}$ is non-singular, the k^{th} pivot is the ratio of the k^{th} leading minor over the $(k-1)^{st}$ leading minor of $\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k$:

$$A^{(k)}_{k,k} \;=\; \frac{\det\overline{(\mathbf{A}^{(k)})_k}}{\det\overline{(\mathbf{A}^{(k)})_{k-1}}} \;=\; \frac{\det\overline{(\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k)_k}}{\det\overline{(\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k)_{k-1}}}.$$

Therefore, in terms of the matrix $\mathbf{AT}_1\mathbf{T}_2\ldots\mathbf{T}_k$, the following column selection rules are equivalent:

(1) given the first $k-1$ columns, select the k^{th} column so as to maximize the absolute value of the resulting k^{th} pivot.

(2) given the first $k-1$ columns, select the k^{th} column so as to maximize the absolute value of the resulting k^{th} leading minor.

3.5 An Alternate Computation of the Pivots

In this Section, we show how to compute the potential pivots through some auxiliary vectors $\boldsymbol{\sigma}_k$ $(1 \le k \le m)$, without explicitly carrying out any LU decomposition.

Lemma 3.5.1 Let $\mathbf{A}$ be a non-singular $m \times m$ matrix. Let β_k $(1 \le k \le m)$ be the sequence of pivots generated by the LU decomposition of $\mathbf{A}$ as described in Section **3.4**. Let $\mathbf{Q} = \mathbf{T}_1\mathbf{T}_2 \ldots \mathbf{T}_{m-1}$ be the permutation matrix resulting from the same LU decomposition. Let $\mathbf{A}' = \mathbf{AQ}$. Let $\begin{pmatrix} \overline{\mathbf{A}'}_{k-1} & \mathbf{a}_k \\ \boldsymbol{\rho}_k & \alpha_k \end{pmatrix}$, where α_k is a scalar, represent the $k \times k$ leading submatrix of $\mathbf{AT}_1\mathbf{T}_2 \ldots \mathbf{T}_k$, and hence the $k \times k$ leading submatrix of $\mathbf{A}'$. Let $K \triangleq \{1 \ldots k\}$.

Then, for $1 \le k \le m$, the m–vector $\boldsymbol{\sigma}_k \triangleq \begin{pmatrix} -\boldsymbol{\rho}_k \overline{\mathbf{A}'}_{k-1}^{-1} & 1 & \mathbf{0} \end{pmatrix}$ is well defined and the pivot β_k is given by the scalar product $\beta_k = \overline{\boldsymbol{\sigma}}_k \; \mathbf{A}'_{Kk}$.

Proof By Corollary **3.4.5**, $\overline{\mathbf{A}'}_{k-1}$ is non-singular and by Corollary **3.4.6**, we have

$$\begin{aligned}
\beta_k &= \frac{\det \begin{pmatrix} \overline{\mathbf{A}'}_{k-1} & \mathbf{a}_k \\ \boldsymbol{\rho}_k & \alpha_k \end{pmatrix}}{\det \overline{\mathbf{A}'}_{k-1}} \\
&= \alpha_k - \boldsymbol{\rho}_k \overline{\mathbf{A}'}_{k-1}^{-1} \mathbf{a}_k \\
&= \begin{pmatrix} -\boldsymbol{\rho}_k \overline{\mathbf{A}'}_{k-1}^{-1} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{a}_k \\ \alpha_k \end{pmatrix} \\
&= \overline{\boldsymbol{\sigma}}_k \; \mathbf{A}'_{Kk}.
\end{aligned}$$

Note that for $1 \le k \le m$, $\beta_k \ne 0$ because $\mathbf{A}$ is non-singular (cf. Corollary **3.4.3**).

The algebraic relationships mentioned in Section **3.4** hold independently of the pivot selection rule. Therefore, at step k, all potential pivots can be computed as products of the form $\overline{\boldsymbol{\sigma}}_k \mathbf{c}$ where $\mathbf{c}$ is one of the unselected columns of $\mathbf{A}_{K\bullet}$, i.e. one of the $m - k + 1$ rightmost columns of $\mathbf{A}'_{K\bullet}$. In other words, the potential pivots are the $m - k + 1$ rightmost elements of the vector $\boldsymbol{\lambda} = \overline{\boldsymbol{\sigma}}_k \mathbf{A}'_{K\overline{K}}$ where $\overline{K} \triangleq \{k \ldots m\}$.

3.6 Factorization with Column Interchanges

If $\mathbf{A}$ is non-singular, the column interchange method described above provides a permutation matrix $\mathbf{Q} = \mathbf{T}_1\mathbf{T}_2 \ldots \mathbf{T}_m$ such that $\mathbf{A}' \triangleq \mathbf{AQ}$ has nonzero leading principal minors. The following theorem explains why the factorization and the column permutation can be carried out simultaneously.

Theorem 3.6.1 Let $\mathbf{A}$ be a non-singular $m \times m$ matrix. Let $\mathbf{Q}$, $\mathbf{A}'$ and $(\boldsymbol{\sigma}_k, \beta_k)$ $(1 \leq k \leq m)$ be as defined in Lemma **3.5.1**. Let α_k $(1 \leq k \leq m)$ denote the diagonal elements of $\mathbf{A}'$. If, for $1 \leq k \leq m$, $\alpha_k \neq 0$, then $\mathbf{A}'$ can be factorized as described in Chapter **2.** and the sequence $(\boldsymbol{\omega}_k, \theta_k)$ $(2 \leq k \leq m)$ resulting from this factorization satisfies

$$\boldsymbol{\omega}_k = \alpha_k^{-1}\boldsymbol{\sigma}_k \quad \text{and} \quad \theta_k = \alpha_k^{-1}\beta_k \quad \text{for } 2 \leq k \leq m \,.$$

Proof The matrix $\mathbf{A}'_{k-1}$ is non-singular and we have

$$\boldsymbol{\sigma}_k \mathbf{A}'_{k-1} = \left(-\boldsymbol{\rho}_k \overline{\mathbf{A}'}_{k-1}^{-1} \quad 1 \quad 0 \right) \begin{pmatrix} \overline{\mathbf{A}'}_{k-1} & 0 & 0 \\ \boldsymbol{\rho}_k & \alpha_k & 0 \\ * & * & * \end{pmatrix} = (\,0 \quad \alpha_k \quad 0\,) = \alpha_k \, \mathbf{u}_k^T.$$

Therefore $\boldsymbol{\sigma}_k \mathbf{A}'_{k-1} = \alpha_k \boldsymbol{\omega}_k \mathbf{A}'_{k-1}$ and $\boldsymbol{\omega}_k = \alpha_k^{-1}\boldsymbol{\sigma}_k$. Finally, we have

$$\theta_k = \det \mathbf{E}_k = \frac{\det \mathbf{A}'_k}{\det \mathbf{A}'_{k-1}} = \frac{\det \begin{pmatrix} \overline{\mathbf{A}'}_{k-1} & \mathbf{a}_k \\ \boldsymbol{\rho}_k & \alpha_k \end{pmatrix}}{\det \begin{pmatrix} \overline{\mathbf{A}'}_{k-1} & 0 \\ \boldsymbol{\rho}_k & \alpha_k \end{pmatrix}} = \frac{\alpha_k - \boldsymbol{\rho}_k \overline{\mathbf{A}'}_{k-1}^{-1} \mathbf{a}_k}{\alpha_k} = \frac{\beta_k}{\alpha_k}.$$

Since $\boldsymbol{\sigma}_k$ can be regarded as the value that $\boldsymbol{\omega}_k$ would take if α_k were equal to one, it can be computed at least as easily as $\boldsymbol{\omega}_k$. As a matter of fact, the method used in Section **2.4** to compute $\boldsymbol{\omega}_k$ entails the computation of $\boldsymbol{\sigma}_k = (\,\boldsymbol{\pi} \quad 1 \quad 0\,)$ where $\boldsymbol{\pi}\overline{\mathbf{A}}_{k-1} = -\boldsymbol{\rho}_k$ is solved by Algorithm **2.3**.

In addition, Theorem **3.6.1** implies that, for $2 \leq k \leq m$, $\theta_k^{-1}\boldsymbol{\omega}_k = \beta_k^{-1}\boldsymbol{\sigma}_k$. Therefore, in Algorithms **2.2** and **2.3**, the sequence $(\boldsymbol{\sigma}_k, \beta_k)$ $(2 \leq k \leq m)$ can replace the sequence $(\boldsymbol{\omega}_k, \theta_k)$ $(2 \leq k \leq m)$ which need not be computed.

Finally, if $\mathbf{A}$ is non-singular then the sequence $(\boldsymbol{\sigma}_k, \beta_k)$ $(1 \leq k \leq m)$ is well defined whereas the existence of $(\boldsymbol{\omega}_k, \theta_k)$ $(2 \leq k \leq m)$ also depends on whether the coefficients α_k are different from zero.

All these considerations lead us to redefine the factorization of $\mathbf{A}'$ in terms of the sequence $(\boldsymbol{\sigma}_k, \beta_k)$ $(1 \leq k \leq m)$. Using the notation of Section **3.5**, we end up with the following method.

```
Algorithm 3.6  (to generate Q and factorize $\mathbf{A}' = \mathbf{AQ}$)
Let $\mathbf{A}' = \mathbf{A}$.
For $k = 1$ to $m$, do the following:
    • Solve $\boldsymbol{\pi}\overline{\mathbf{A}'}_{k-1} = -\boldsymbol{\rho}_k$.
    • Let $\boldsymbol{\sigma}_k = (\boldsymbol{\pi} \quad 1 \quad \mathbf{0})$.
    • Compute the vector of potential pivots $\boldsymbol{\lambda} = \overline{\boldsymbol{\sigma}}_k \mathbf{A}'_{K\overline{K}}$.
    • Select a pivot column $j_k$ such that $|\lambda_{j_k}| = \max_{k \leq j \leq m} |\lambda_j|$.
    • Let $\mathbf{A}' = \mathbf{A}'\mathbf{T}_{k,j_k}$.
    • Let $\beta_k = \lambda_{j_k}$.
    • If $\beta_k = 0$ then stop ($\mathbf{A}$ is singular).
```

Example: Consider the 3×3 matrix introduced in Section **1.2**:

$$\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}.$$

- Step 1 Column 1 has the entry of largest absolute value in row 1. Therefore, column 1 remains in first position.

$$\boldsymbol{\sigma}_1 = (1 \quad * \quad *)$$

$$\boldsymbol{\lambda} = (1)\,(8 \quad 1 \quad 6) = (8 \quad 1 \quad 6)$$

$$\beta_1 = 8$$

- Step 2 We compute the vector $\boldsymbol{\sigma}_2$:

$$\boldsymbol{\sigma}_2 \begin{pmatrix} 8 & 0 \\ 3 & 1 \end{pmatrix} = (0 \quad 1)$$

$$\boldsymbol{\sigma}_2 = (\boldsymbol{\pi} \quad 1 \quad *) \quad \text{where} \quad \boldsymbol{\pi}\,(8) = (-3)$$

$$\boldsymbol{\sigma}_2 = (-\tfrac{3}{8} \quad 1 \quad *)$$

The potential pivots are given by

$$\boldsymbol{\lambda} = (-\tfrac{3}{8} \quad 1)\begin{pmatrix} 1 & 6 \\ 5 & 7 \end{pmatrix} = (\tfrac{37}{8} \quad \tfrac{38}{8}).$$

We select $\beta_2 = \frac{38}{8} = \frac{19}{4}$ from column 3 which moves into second position.

- Step 3 We compute the vector $\boldsymbol{\sigma}_3$:

$$\boldsymbol{\sigma}_3 \begin{pmatrix} 8 & 6 & 0 \\ 3 & 7 & 0 \\ 4 & 2 & 1 \end{pmatrix} = (0 \quad 0 \quad 1)$$

$$\boldsymbol{\sigma}_3 = (\boldsymbol{\pi} \quad 1) \quad \text{where} \quad \boldsymbol{\pi}\begin{pmatrix} 8 & 6 \\ 3 & 7 \end{pmatrix} = (-4 \quad -2)$$

$$\overline{\boldsymbol{\pi}}_1 = (-\tfrac{1}{2} \quad *)$$

$$\overline{\boldsymbol{\pi}}_2 = (-\tfrac{1}{2} \quad *) + \left[-2 - (-\tfrac{1}{2} \quad *)\begin{pmatrix} 6 \\ * \end{pmatrix}\right] \tfrac{4}{19}(-\tfrac{3}{8} \quad 1) = (-\tfrac{11}{19} \quad \tfrac{4}{19})$$

$$\boldsymbol{\sigma}_3 = (-\tfrac{11}{19} \quad \tfrac{4}{19} \quad 1)$$

The only remaining pivot is given by

$$\boldsymbol{\lambda} = (-\tfrac{11}{19} \quad \tfrac{4}{19} \quad 1)\begin{pmatrix} 1 \\ 5 \\ 9 \end{pmatrix} = \frac{180}{19}$$

$$\beta_3 = \tfrac{180}{19}$$

- Summary

$$\mathbf{A}' = \mathbf{AQ} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 8 & 6 & 1 \\ 3 & 7 & 5 \\ 4 & 2 & 9 \end{pmatrix}$$

$$\mathbf{A}'_1 = \begin{pmatrix} 8 & * & * \\ 3 & 7 & * \\ 4 & 2 & 9 \end{pmatrix} \qquad \boldsymbol{\sigma}_1 = (\,1 \quad * \quad *\,) \qquad \beta_1 = 8$$

$$\mathbf{v}'_2 = \begin{pmatrix} 6 \\ * \\ * \end{pmatrix}$$

$$\mathbf{A}'_2 = \begin{pmatrix} 8 & 6 & * \\ 3 & 7 & * \\ 4 & 2 & 9 \end{pmatrix} \qquad \boldsymbol{\sigma}_2 = (\,-\tfrac{3}{8} \quad 1 \quad *\,) \qquad \beta_2 = \tfrac{19}{4}$$

$$\mathbf{v}'_3 = \begin{pmatrix} 1 \\ 5 \\ * \end{pmatrix}$$

$$\mathbf{A}'_3 = \begin{pmatrix} 8 & 6 & 1 \\ 3 & 7 & 5 \\ 4 & 2 & 9 \end{pmatrix} \qquad \boldsymbol{\sigma}_3 = (\,-\tfrac{11}{19} \quad \tfrac{4}{19} \quad 1\,) \qquad \beta_3 = \tfrac{180}{19}$$

3.7 Rescaling of the Matrix A

To reduce the number of divisions during the factorization of a matrix, it may be worthwhile to rescale the columns of the matrix so that the resulting pivots β_k $(1 \leq k \leq m)$ are equal to one. For this rescaling to speed up the factorization, the number nz of nonzero elements off the diagonal must be less than the number of divisions by β_k :

$$nz < \sum_{k=1}^{m}(m-k) = \frac{m(m-1)}{2}.$$

Therefore, the density of the matrix must be less than $\frac{1}{2}$. This procedure does not affect the number of divisions involved in solving the system $\mathbf{Ax} = \mathbf{b}$ for instance, because of the unscaling of $\mathbf{x}$.

Example:

Consider the 3×3 matrix introduced in Section **1.2** $\mathbf{A} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}$.

After the example of Section **3.6**, it is easy to see that the matrix

$$\mathbf{A}'' = \mathbf{AQS} = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{8} & 0 & 0 \\ 0 & \frac{4}{19} & 0 \\ 0 & 0 & \frac{19}{180} \end{pmatrix} = \begin{pmatrix} 1 & \frac{24}{19} & \frac{19}{180} \\ \frac{3}{8} & \frac{28}{19} & \frac{95}{180} \\ \frac{1}{2} & \frac{8}{19} & \frac{19}{20} \end{pmatrix}$$

has all its pivots equal to one.

$$\mathbf{A}_1'' = \begin{pmatrix} 1 & * & * \\ \frac{3}{8} & \frac{28}{19} & * \\ \frac{1}{2} & \frac{8}{19} & \frac{19}{20} \end{pmatrix} \qquad \boldsymbol{\sigma}_1 = (\, 1 \quad * \quad * \,)$$

$$\mathbf{v}_2'' = \begin{pmatrix} \frac{24}{19} \\ * \\ * \end{pmatrix}$$

$$\mathbf{A}_2'' = \begin{pmatrix} 1 & \frac{24}{19} & * \\ \frac{3}{8} & \frac{28}{19} & * \\ \frac{1}{2} & \frac{8}{19} & \frac{19}{20} \end{pmatrix} \qquad \boldsymbol{\sigma}_2 = (\, -\frac{3}{8} \quad 1 \quad * \,)$$

$$\mathbf{v}_3'' = \begin{pmatrix} \frac{19}{180} \\ \frac{95}{180} \\ * \end{pmatrix}$$

$$\mathbf{A}_3'' = \begin{pmatrix} 1 & \frac{24}{19} & \frac{19}{180} \\ \frac{3}{8} & \frac{28}{19} & \frac{95}{180} \\ \frac{1}{2} & \frac{8}{19} & \frac{19}{20} \end{pmatrix} \qquad \boldsymbol{\sigma}_3 = (\, -\frac{11}{19} \quad \frac{4}{19} \quad 1 \,)$$

CHAPTER 4. SPIKE REDUCTION

4.1 Introduction

The method presented in Section **2.5** to factorize a matrix $\mathbf{A}$ and to solve a system $\mathbf{Ax} = \mathbf{b}$ or $\boldsymbol{\pi}\mathbf{A} = \boldsymbol{\gamma}$ will work better if the number of auxiliary vectors $\boldsymbol{\omega}_k$ (or $\boldsymbol{\sigma}_k$) is smaller, and, given their number, if their size is smaller.

The most obvious way to achieve these goals is to reduce the number of spikes of the matrix $\mathbf{A}$ and, whenever possible, to shift those spikes towards the left. To that end, we describe three myopic algorithms, inspired by Hellerman and Rarick's P^3 procedure (Hellerman and Rarick, 1972), that reorder the matrix by selecting some rows and columns, assigning them a position, deleting them and repeating the process on the resulting submatrix until all rows or all columns have been assigned.

4.2 Definitions and Notation

Let i and j denote a row and a column index of $\mathbf{A}$. If $A_{ij} \neq 0$, we say that row i "intersects" column j, that column j "intersects" row i or that row i and column j "intersect". We define the following:

ROWSPAN(i)	the set of columns j intersecting row i.
COLSPAN(j)	the set of rows i intersecting column j.
ROWCOUNT(i)	the number of nonzero entries in row i.
COLCOUNT(j)	the number of nonzero entries in column j.
ROWTALLY(i)	the number of nonzero entries in the columns intersecting row i.
COLTALLY(j)	the number of nonzero entries in the rows intersecting column j.

$$
\begin{aligned}
\text{ROWSPAN}(i) &\triangleq \{j : A_{ij} \neq 0\} \\
\text{COLSPAN}(j) &\triangleq \{i : A_{ij} \neq 0\} \\
\text{ROWCOUNT}(i) &\triangleq |\text{ROWSPAN}(i)| \\
\text{COLCOUNT}(j) &\triangleq |\text{COLSPAN}(j)| \\
\text{ROWTALLY}(i) &\triangleq \sum_{j:A_{ij}\neq 0} \text{COLCOUNT}(j) \\
\text{COLTALLY}(j) &\triangleq \sum_{i:A_{ij}\neq 0} \text{ROWCOUNT}(i)
\end{aligned}
$$

A line of a matrix **A** is a row or a column of **A**. At iteration k, the "active submatrix" is the submatrix obtained after deletion of the lines selected during iterations $1, \ldots, k-1$.

4.3 Top-Left Spike Reduction Algorithm

This algorithm essentially keeps selecting from the active submatrix a row to be placed at the top of the unassigned rows and a matching column to be placed to the left of the unassigned columns. This amounts to identifying the coefficient of the active submatrix to be placed in the top-left corner.

More precisely, at iteration k, a row with the fewest nonzero entries is selected from the active submatrix and assigned position k (the highest available). If some columns of the active submatrix intersect that row, one of them is assigned position k (the leftmost available) and the others are sent to a spike-index queue. Otherwise, a column is removed from the spike-index queue according to the First-In-First-Out (FIFO) priority rule and assigned the position k. Finally, the active submatrix is updated by deletion of the selected row and of the columns intersecting it.

Termination occurs when all columns have been deleted. Then the undeleted rows are assigned the bottom positions, the columns remaining in the spike-index queue are removed in FIFO order and assigned the rightmost positions. Under this algorithm, deletion of all rows cannot occur before deletion of all columns.

In addition, note that the spikes added to the queue during iteration k will have zero entries above row k and a nonzero entry in row k of the current matrix, and that the only unassigned columns with nonzero elements above row k are already in the spike queue. Therefore, if no column intersects the selected row and no spike is available in the queue at iteration k, then the first k rows of the reordered matrix contain only $k - 1$ nonzero columns and the matrix $\mathbf{A}$ is structurally singular (i.e. singular for any values given to the nonzero coefficients).

Finally, the FIFO priority rule used in removing the indices from the spike-index queue produces spikes in order of non-increasing heights, which prevents the structural singularity of the leading submatrices, unless the whole matrix $\mathbf{A}$ is itself structurally singular.

The Top-Left spike reduction algorithm is listed below:

```
Let $k = 0$.
Repeat
    Let $k = k + 1$;
    TOPLEFT($k$);
until all columns are deleted.
Assign undeleted rows positions $\{k + 1 \ldots m\}$.
Remove columns remaining in spike-index queue (FIFO).
Assign these columns positions $\{k + 1 \ldots m\}$.
```

The procedure TOPLEFT(k) consists of the following instructions:

- Select from the active submatrix a row i_k minimizing ROWCOUNT(i). Break ties by maximizing ROWTALLY(i).
- Assign row i_k position k.
- If ROWCOUNT(i_k) $\neq 0$, then

```
    select a pivot column $j_k$ from the columns intersecting row $i_k$;
    assign column $j_k$ position $k$;
    add the other columns intersecting row $i_k$ to the spike-index queue.
```

- If ROWCOUNT(i_k) = 0, then

  remove the FIFO column from the spike queue and assign it position k.
- Delete row i_k and the columns intersecting row i_k.

4.4 Bottom-Right Spike Reduction Algorithm

Instead of building the new matrix from the top-left corner to the bottom-right one, the second algorithm does it in reverse direction, skipping some spaces for the spikes. It essentially keeps selecting from the active submatrix a column to be placed to the right of the unassigned columns and a matching row to be placed at the bottom of the unassigned rows. This amounts to identifying the coefficient of the active submatrix to be placed in the bottom-right corner.

At each iteration, a column with the fewest nonzero entries is selected from the active submatrix. If some rows of the submatrix intersect that column, they are assigned the positions at the bottom of the submatrix, the selected column is assigned the rightmost position p allowing it to fit on and below the diagonal, and the unassigned positions to the right of p are sent to a spike-position queue. Otherwise, a position is removed from the spike-position queue according to the First-In-First-Out (FIFO) priority rule and the selected column is assigned that position. Finally, the active submatrix is updated by deletion of the selected column and of the rows intersecting it.

Termination occurs when all rows have been deleted. Then the undeleted columns are assigned the positions remaining in the spike-position queue. Note that, under this algorithm, deletion of all columns cannot occur before deletion of all rows.

If no row intersects the selected column and no spike-position is available in the queue at iteration k, then some l columns of $\mathbf{A}$ contain at most $l-1$ nonzero rows and the matrix $\mathbf{A}$ is structurally singular.

The Bottom-Right algorithm obtains the spikes in order of non-decreasing heights, but it allocates them from right to left. However, the Bottom-Right algorithm, listed below, usually yields fewer spikes than the Top-Left algorithm:

```
Let $k = m + 1$.
Let $p_k = m + 1$.
Repeat
    Let $k = k - 1$;
    BOTTOMRIGHT($k$);
until all rows are deleted.
Assign the undeleted columns the spike-positions remaining in queue.
```

The procedure BOTTOMRIGHT(k) consists of the following instructions:

- Select from the active submatrix a column j_k minimizing COLCOUNT(j); break eventual tie by maximizing COLTALLY(j).
- Update the rightmost non-spike position $p_k = p_{k+1} -$ COLCOUNT(j).
- If COLCOUNT(j_k) $\neq 0$, then

    assign column j_k position p_k;

    select a pivot row from the rows intersecting column j_k;

    assign the pivot row position p_k;

    assign the other rows intersecting column j_k the positions in the range $p_k + 1 \ldots p_{k+1} - 1$;

    add $p_k + 1, \ldots, p_{k+1} - 1$ to the spike-position queue.
- If COLCOUNT(j_k) $= 0$, then

    remove the FIFO position q from the spike-position queue;

    assign column j_k position q.
- Delete column j_k and the rows intersecting column j_k.

4.5 Composite Spike Reduction Algorithm

The third algorithm is a combination of the first two. At odd iterations, it selects and deletes a column and the rows intersecting it (using the procedure BOTTOMRIGHT). At even iterations, it selects and deletes a row and the columns intersecting it (using the procedure TOPLEFT).

Termination occurs when all rows have been deleted. Then, the undeleted columns are added to the spike-index queue. Finally the columns remaining in the spike-index queue are matched with the positions remaining in the spike-position queue.

Empirically, this algorithm appears to combine the speed of the first one (i.e. it requires a small number of iterations) and the efficiency of the second one (i.e. it yields a small number of spikes).

The composite spike reduction algorithm is listed below:

```
Let $k = m + 1$.
Let $p_k = m + 1$.
Let $k' = 0$.
Repeat
    If not all columns are deleted then
        Let $k = k - 1$;
        BOTTOMRIGHT($k$).
    If not all rows are deleted then
        Let $k' = k' + 1$;
        TOPLEFT($k'$).
until all rows are deleted.
Add undeleted columns to the spike-index queue.
Match remaining spike-indices and remaining spike-positions.
```

4.6 Examples

In this paragraph, we apply the three algorithms described above to an 8×8 matrix. Although the numerical values of the nonzero coefficients are not needed, they are represented for the sake of consistency with the example used in Chapter 4. SI queue and SP queue denote the spike-index queue and the spike-position queue respectively. The columns belonging to the SI queue are printed in italic.

4.6.1 First algorithm:

	1	2	3	4	5	6	7	8	count	tally
1	8.00	9.00	5.00	1.00				-7.00	5	
2		3.00				8.00	2.00		3	
3				**6.00**	**4.00**				**2**	
4	-3.00	5.00		-9.00				2.00	4	
5		4.00	-7.00	3.00					3	
6					-2.00	5.00	-1.00		3	
7				5.00	-8.00	2.00	6.00	-4.00	5	
8		5.00	2.00					3.00	3	
count	2	5	3	5	3	3	3	4		

Matrix 0

Iteration 1

Row 3 is selected and assigned position 1.

Columns 4 and 5 are selected.

Column 4 is assigned position 1 and column 5 is sent to the SI queue.

	4	1	2	3	6	7	8	5	count	tally
3	6.00							*4.00*		
1	1.00	8.00	9.00	5.00			-7.00		4	
2			3.00		8.00	2.00			3	
4	-9.00	-3.00	5.00				2.00		3	
5	3.00		**4.00**	**-7.00**					**2**	**8**
6					5.00	-1.00		*-2.00*	2	6
7	5.00				2.00	6.00	-4.00	*-8.00*	3	
8			5.00	2.00			3.00		3	
count		2	5	3	3	3	4			

Matrix 1a

Iteration 2

Row 5 is selected and assigned position 2.

Columns 2 and 3 are selected.

Column 3 is assigned position 2 and column 2 joins column 5 in the SI queue.

	4	3	1	6	7	8	5	2	count	tally
3	6.00						4.00			
5	3.00	-7.00						4.00		
1	1.00	5.00	8.00			-7.00		9.00	2	
2				8.00	2.00			3.00	2	
4	-9.00		-3.00			2.00		5.00	2	
6				5.00	-1.00		-2.00		2	
7	5.00			2.00	6.00	-4.00	-8.00		3	
8		2.00				3.00		5.00	1	
count			2	3	3	4				

Matrix 2a

Iteration 3

Row 8 is selected and assigned position 3.

Column 8 is selected and assigned position 3.

	4	3	8	1	6	7	5	2	count	tally
3	6.00						4.00			
5	3.00	-7.00						4.00		
8		2.00	3.00					5.00		
1	1.00	5.00	-7.00	8.00				9.00	1	
2					8.00	2.00		3.00	2	
4	-9.00		2.00	-3.00				5.00	1	
6					5.00	-1.00	-2.00		2	
7	5.00		-4.00		2.00	6.00	-8.00		2	
count				2	3	3				

Matrix 3a

Iteration 4

Row 1 is selected and assigned position 4.

Column 1 is selected and assigned position 4.

	4	3	8	1	6	7	5	*2*	count	tally
3	6.00						*4.00*			
5	3.00	-7.00						*4.00*		
8		2.00	3.00					*5.00*		
1	1.00	5.00	-7.00	8.00				*9.00*		
2					8.00	2.00		*3.00*	2	
4	-9.00		2.00	-3.00				*5.00*	0	
6					5.00	-1.00	*-2.00*		2	
7	5.00		-4.00		2.00	6.00	*-8.00*		2	
count					3	3				

Matrix 4a

Iteration 5

Row 4 is selected and assigned position 5.

Column 5 is removed from the bottom of the SI queue and assigned position 5.

	4	3	8	1	5	6	7	*2*	count	tally
3	6.00				4.00					
5	3.00	-7.00						*4.00*		
8		2.00	3.00					*5.00*		
1	1.00	5.00	-7.00	8.00				*9.00*		
4	-9.00		2.00	-3.00				*5.00*		
2						**8.00**	**2.00**	*3.00*	**2**	**6**
6					-2.00	5.00	-1.00		2	6
7	5.00		-4.00		-8.00	2.00	6.00		2	6
count						3	3			

Matrix 5a

Iteration 6

Row 2 is selected and assigned position 6.

Columns 6 and 7 are selected.

Column 6 is assigned position 6 and column 7 joins column 2 in the SI queue.

	4	3	8	1	5	6	2	7	count	tally
3	6.00				4.00					
5	3.00	-7.00					*4.00*			
8		2.00	3.00				*5.00*			
1	1.00	5.00	-7.00	8.00			*9.00*			
4	-9.00		2.00	-3.00			*5.00*			
2						8.00	*3.00*	*2.00*		
6					-2.00	5.00		*-1.00*	0	
7	5.00		-4.00		-8.00	2.00		*6.00*	0	
count										

Matrix 6a

After 6 iterations, all columns have been deleted. The remaining rows 6 and 7 are assigned the remaining positions 7 and 8, as are the columns 2 and 7 left in the SI queue. The final matrix has three spikes, namely columns 5, 2 and 7 in positions 5, 7 and 8 respectively. By interchanging columns 5 and 7, we would obtain only two spikes but the resulting 5×5 leading submatrix would be structurally singular. The FIFO rule prevents such occurences when the whole matrix is non-singular.

4.6.2 Second algorithm:

	1	2	3	4	5	6	7	8	count
1	**8.00**	9.00	5.00	1.00				-7.00	5
2		3.00				8.00	2.00		3
3				6.00	4.00				2
4	**-3.00**	5.00		-9.00				2.00	4
5		4.00	-7.00	3.00					3
6					-2.00	5.00	-1.00		3
7				5.00	-8.00	2.00	6.00	-4.00	5
8		5.00	2.00					3.00	3
count	2	5	3	5	3	3	3	4	
tally									

Matrix 0

Iteration 1

Column 1 is selected and assigned position $p = 9 - 2 = 7$.

Rows 1 and 4 are selected and assigned positions 7 and 8.

Position 8 is sent to the SP queue.

	2	3	4	5	6	7	1	8	count
2	3.00				8.00	2.00			3
3			6.00	4.00					2
5	4.00	-7.00	3.00						3
6				-2.00	5.00	-1.00			3
7			5.00	-8.00	2.00	6.00		**-4.00**	5
8	5.00	2.00						**3.00**	3
1	9.00	5.00	1.00				8.00	-7.00	
4	5.00		-9.00				-3.00	2.00	
count	3	2	3	3	3	3		**2**	
tally		6						**8**	

Matrix 1b

Iteration 2

Column 8 is selected and assigned position $p = 7 - 2 = 5$.

Rows 7 and 8 are selected and assigned positions 5 and 6.

Position 6 joins 8 in the SP queue.

	2	3	4	5	8	7	1	6	count
2	3.00					2.00		8.00	3
3			6.00	4.00					2
5	4.00	-7.00	3.00						3
6				-2.00		-1.00		5.00	3
7			5.00	-8.00	-4.00	6.00		2.00	
8	5.00	2.00			3.00				
1	9.00	5.00	1.00		-7.00		8.00		
4	5.00		-9.00		2.00		-3.00		
count	2	1	2	2		2		2	
tally									

Matrix 2b

Iteration 3

Column 3 is selected and assigned position $p = 5 - 1 = 4$.

Row 5 is selected and assigned position 4.

	2	4	5	3	8	7	1	6	count
2	3.00					2.00		8.00	3
3		6.00	4.00						2
6			-2.00			-1.00		5.00	3
5	4.00	3.00		-7.00					
7		5.00	-8.00		-4.00	6.00		2.00	
8	5.00			2.00	3.00				
1	9.00	1.00		5.00	-7.00		8.00		
4	5.00	-9.00			2.00		-3.00		
count	1	1	2			2		2	
tally	3	2							

Matrix 3b

Iteration 4

Column 2 is selected and assigned position $p = 4 - 1 = 3$.

Row 2 is selected and assigned position 3.

	4	5	2	3	8	7	1	6	count
3	6.00	4.00							2
6		-2.00				-1.00		5.00	3
2			3.00			2.00		8.00	
5	3.00		4.00	-7.00					
7	5.00	-8.00			-4.00	6.00		2.00	
8			5.00	2.00	3.00				
1	1.00		9.00	5.00	-7.00		8.00		
4	-9.00		5.00		2.00		-3.00		
count	1	2				1		1	
tally	2					3		3	

Matrix 4b

Iteration 5

Column 6 is selected and assigned position $p = 3 - 1 = 2$.

Row 6 is selected and assigned position 2.

	4	6	2	3	8	5	1	**7**	count
3	$\boxed{6.00}$					$\boxed{4.00}$		$\boxed{}$	2
6		5.00				-2.00		-1.00	
2		8.00	3.00					2.00	
5	3.00		4.00	-7.00					
7	5.00	2.00			-4.00	-8.00		6.00	
8			5.00	2.00	3.00				
1	1.00		9.00	5.00	-7.00		8.00		
4	-9.00		5.00		2.00		-3.00		
count	1					1		**0**	
tally									

Matrix 5b

Iteration 6

Column 7 is selected and assigned position $q = 8$.

Position 8 is removed from the SP queue.

	4	6	2	3	8	5	1	7	count
3	$\boxed{6.00}$					$\boxed{4.00}$			2
6		5.00				-2.00		-1.00	
2		8.00	3.00					2.00	
5	3.00		4.00	-7.00					
7	5.00	2.00			-4.00	-8.00		6.00	
8			5.00	2.00	3.00				
1	1.00		9.00	5.00	-7.00		8.00		
4	-9.00		5.00		2.00		-3.00		
count	**1**					1			
tally	**2**					2			

Matrix 6b

Iteration 7

Column 4 is selected and assigned position $p = 2 - 1 = 1$.

Row 3 is selected and assigned position 1.

	4	6	2	3	8	5	1	7	count
3	6.00					$\overline{4.00}$			
6		5.00				-2.00		-1.00	
2		8.00	3.00					2.00	
5	3.00		4.00	-7.00					
7	5.00	2.00			-4.00	-8.00		6.00	
8			5.00	2.00	3.00				
1	1.00		9.00	5.00	-7.00		8.00		
4	-9.00		5.00		2.00		-3.00		
count						0			
tally									

Matrix 7b

After 7 iterations, all rows have been deleted. The remaining column 5 is matched with the position remaining in the SP queue. The final matrix has only two spikes, namely columns 5 and 7, in positions 6 and 8 respectively.

4.6.3 Third algorithm:

	1	2	3	4	5	6	7	8	count	tally
1	**8.00**	9.00	5.00	1.00				-7.00	5	
2		3.00				8.00	2.00		3	
3				6.00	4.00				2	
4	**-3.00**	5.00		-9.00				2.00	4	
5		4.00	-7.00	3.00					3	
6					-2.00	5.00	-1.00		3	
7				5.00	-8.00	2.00	6.00	-4.00	5	
8		5.00	2.00					3.00	3	
count	2	5	3	5	3	3	3	4		
tally										

Matrix 0

Iteration 1

Column 1 is selected and assigned position $p = 9 - 2 = 7$.

Rows 1 and 4 are selected and assigned positions 7 and 8.

Position 8 is sent to the SP queue.

	2	3	4	5	6	7	1	8	count	tally
2	3.00				8.00	2.00			3	
3			**6.00**	**4.00**					**2**	
5	4.00	-7.00	3.00						3	
6				-2.00	5.00	-1.00			3	
7			5.00	-8.00	2.00	6.00		-4.00	5	
8	5.00	2.00						3.00	3	
1	9.00	5.00	1.00				8.00	-7.00		
4	5.00		-9.00				-3.00	2.00		
count	3	2	3	3	3	3		2		
tally										

Matrix c1

Iteration 2

Row 3 is selected and assigned position 1.

Columns 4 and 5 are selected.

Column 4 is assigned position 1 and column 5 sent to the SI queue.

	4	2	3	6	7	**8**	1	5	count	tally
3	6.00							*4.00*		
2		3.00		8.00	2.00				3	
5	3.00	4.00	-7.00						2	
6				5.00	-1.00			*-2.00*	2	
7	5.00			2.00	6.00	**-4.00**		*-8.00*	3	
8		5.00	2.00			**3.00**			3	
1	1.00	9.00	5.00			-7.00	8.00			
4	-9.00	5.00				2.00	-3.00			
count		3	2	3	3	**2**				
tally			5			**6**				

Matrix c2

Iteration 3

Column 8 is selected and assigned position $p = 7 - 2 = 5$.

Rows 7 and 8 are selected and assigned positions 5 and 6.

Position 6 is sent to the SP queue.

	4	2	3	6	8	7	1	5	count	tally
3	6.00							*4.00*		
2		3.00		8.00		2.00			3	
5	3.00	4.00	-7.00						2	3
6				5.00		-1.00		*-2.00*	2	4
7	5.00			2.00	-4.00	6.00		*-8.00*		
8		5.00	2.00		3.00					
1	1.00	9.00	5.00		-7.00		8.00			
4	-9.00	5.00			2.00		-3.00			
count		2	1	2		2				
tally										

Matrix c3

Iteration 4

Row 6 is selected and assigned position 2.

Columns 6 and 7 are selected.

Column 6 is assigned position 2 and column 7 is added to the SI queue.

	4	6	2	3	8	5	1	7	count	tally
3	6.00					*4.00*				
6		5.00				*-2.00*		*-1.00*		
2		8.00	3.00					*2.00*	1	
5	3.00		4.00	-7.00					2	
7	5.00	2.00			-4.00	*-8.00*		*6.00*		
8			5.00	2.00	3.00					
1	1.00		9.00	5.00	-7.00		8.00			
4	-9.00		5.00		2.00		-3.00			
count			2	1						
tally										

Matrix c4

Iteration 5

Column 3 is selected and assigned position $p = 5 - 1 = 4$.

Row 5 is selected and assigned position 4.

	4	6	2	3	8	5	1	7	count	tally
3	6.00					*4.00*				
6		5.00				*-2.00*		*-1.00*		
2		8.00	$\boxed{3.00}$					*2.00*	1	
5	3.00		4.00	-7.00						
7	5.00	2.00			-4.00	*-8.00*		*6.00*		
8			5.00	2.00	3.00					
1	1.00		9.00	5.00	-7.00		8.00			
4	-9.00		5.00		2.00		-3.00			
count			1							
tally										

Matrix c5

Iteration 6

Row 2 is selected and assigned position 3.

Column 2 is selected and assigned position 3.

	4	6	2	3	8	5	1	7	count	tally
3	6.00					*4.00*				
6		5.00				*-2.00*		*-1.00*		
2		8.00	3.00					*2.00*		
5	3.00		4.00	-7.00						
7	5.00	2.00			-4.00	*-8.00*		*6.00*		
8			5.00	2.00	3.00					
1	1.00		9.00	5.00	-7.00		8.00			
4	-9.00		5.00		2.00		-3.00			
count										
tally										

Matrix c6

After 6 iterations, all rows and all columns have been deleted. By matching the spike-index queue {5,7} with the spike-position queue {8,6} in reverse order, we obtain the same final matrix as with the second algorithm. The two spikes, columns 5 and 7, are in positions 6 and 8 respectively. In this particular example, the third algorithm takes as few iterations as the first one and yields as few spikes as the second one.

CHAPTER 5. FACTORIZATION ALGORITHM

5.1 Introduction

The spike reduction algorithms described in Chapter **4** do not take into consideration the numerical values of the nonzero coefficients of the matrix $\mathbf{A}$. Therefore, they are unlikely to produce a matrix whose direct factorization would be stable. On the other hand, the factorization with column interchanges described in Chapter **3** would result in too many spikes and too much computation if it were applied to the original matrix $\mathbf{A}$.

The algorithm described in this chapter attempts to create a sparse and stable factorization by combining the ideas described in the previous three chapters. It consists of three phases: prescaling, preordering and factorization.

The prescaling phase is a simple column scaling. The preordering phase is based on the composite spike minimization algorithm of Section **4.5**. The factorization phase, which also includes some reordering and rescaling, is a restricted-pivoting version of the factorization with column interchanges described in Section **3.6**.

5.2 Notation

We use the definitions and notation of Section **4.2**. In addition, we introduce a pivot tolerance τ and the following quantities:

$$\textsc{rowpivot}(i) \triangleq \max_k |A_{ik}|$$

$$\textsc{colpivot}(j) \triangleq \max_k |A_{kj}|$$

$$\textsc{rowscore}(i) \triangleq \begin{cases} \textsc{rowpivot}(i) \times \textsc{rowtally}(i) & \text{if } \textsc{rowpivot}(i) > \tau \\ 0 & \text{otherwise} \end{cases}$$

$$\textsc{colscore}(j) \triangleq \begin{cases} \textsc{colpivot}(j) \times \textsc{coltally}(j) & \text{if } \textsc{colpivot}(j) > \tau \\ 0 & \text{otherwise} \end{cases}$$

Thus, if a row i satisfies $\textsc{rowscore}(i) = 0$, it contains no entry that can be accepted as a pivot.

5.3 Prescaling Phase

To render meaningful any row or column selection based on the size of the coefficients within a column or a row of **A**, these coefficients of **A** must have been prescaled. Here, we assume a columnwise matrix representation and we choose to scale the columns of **A**. For instance, we can divide each column by a coefficient of largest absolute value in that column. The resulting columns are unit vectors for the $\| \cdot \|_\infty$ norm.

5.4 Preordering Phase

The preordering is aimed at reducing the number and size of the spikes and, hence, of the factors. It uses the composite spike reduction algorithm of Section **4.5** with three modifications.

The first modification concerns row or column selection. The selected rows and columns that do not contain a suitable pivot (i.e. an entry whose absolute value is greater than τ) are rejected. When a column has been selected and several rows are available to be assigned the pivot position, the row containing the entry of largest absolute value in that column inside the active submatrix is chosen as pivot row, unless that entry remains too small to be used as a pivot in which case the selected column becomes a spike. Similarly, when a row has been selected and several columns are available to be assigned the pivot position, the column containing the entry of largest absolute value in that row inside the active submatrix is chosen as pivot column, unless that entry remains too small to be used as a pivot in which case the pivot position is sent to the spike-position queue and all the columns intersecting the selected row are sent to the spike-index queue.

The second modification concerns the tiebreaking function TALLY() that is replaced by the "multiobjective" function SCORE() to take into account both the pivot size and the number of nonzero entries to be deleted.

Finally, the third modification concerns the ordering of the spikes. Spikes are still given in order of nonincreasing height, but their final order is to be determined in the factorization phase. Therefore, spike-indices and spike-positions are only added to, but not removed from their respective queues.

5.5 Factorization Phase

The preordering phase induces a partition of the columns into spikes and non-spikes. The factorization phase preserves the ordering of the rows and that of the non-spikes but permutes the spikes according to the pivot-maximizing rule introduced in Section **3.6**.

For all spike-positions k in increasing order (i.e. from left to right), it computes a vector $\overline{\boldsymbol{\sigma}}_k$ and the potential pivots associated with the available spikes, selects a pivot of largest absolute value and matches the corresponding spike-index with the spike-position k. If no suitable pivot can be obtained from the spike-index queue, the matrix is deemed singular and the algorithm stops.

In addition, each time a spike has been assigned a position, it may be rescaled so that the resulting pivot takes the value 1. This step is not recommended for dense matrices (cf. Section **3.7**).

5.6 Factorization Algorithm

All the procedures mentioned above, including the column rescaling, are contained in the following algorithm:

```
Normalize the columns of $\mathbf{A}$.
Let $k = m + 1$.
Let $p_k = m + 1$.
Let $k' = 0$.
Repeat
    If not all columns are deleted then
        Let $k = k - 1$;
        BOTTOMRIGHT($k$).
    If not all rows are deleted then
        Let $k' = k' + 1$;
        TOPLEFT($k'$).
until all rows are deleted.
Add undeleted columns to the spike-index queue.
MATCHQUEUES.
```

The procedure BOTTOMRIGHT(k) becomes:

- Select from the active submatrix a column j_k minimizing COLCOUNT(j); break eventual tie by maximizing COLSCORE(j).
- Update the rightmost non-spike position $p_k = p_{k+1} -$ COLCOUNT(j).
- If COLSCORE(j_k) $\neq 0$, then

    assign column j_k position p_k;

    select a pivot row from the rows intersecting column j_k;

    assign the pivot row position p_k;

    assign the other rows intersecting column j_k the positions in the range $p_k + 1, \ldots, p_{k+1} - 1$;

    add the positions $p_k + 1 \ldots p_{k+1} - 1$ to the spike-position queue;

    rescale the pivot column so that the resulting pivot equals 1.
- If COLSCORE(j_k) $= 0$, then

    add column j_k to the spike-index queue.
- Delete column j_k and the rows intersecting column j_k.

The procedure TOPLEFT(k) becomes:

```
• Select from the active submatrix a row $i_k$ minimizing ROWCOUNT($i$) ; break
  eventual tie by maximizing ROWSCORE($i$).
• Assign row $i_k$ position $k$.
• If ROWSCORE($i_k$) $\neq 0$, then
      select a pivot column $j_k$ from the columns intersecting row $i_k$;
      assign column $j_k$ position $k$;
      add the other columns intersecting row $i_k$ to the spike-index queue.
• If ROWSCORE($i_k$) $= 0$ then
      add position $k$ to spike-position queue;
      add all columns intersecting row $i_k$ to spike-index queue.
• Delete row $i_k$ and the columns intersecting row $i_k$.
```

The procedure MATCHQUEUES stands for:

```
• While spike-position queue $\neq \emptyset$,
      remove leftmost position $k$ from spike-position queue;
      compute $\overline{\boldsymbol{\sigma}}_k$;
      compute the potential pivots associated with the columns of the spike-
      index queue;
      select pivot $\beta_k$ and pivot-maximizing column $l$;
      if $|\beta_k| < \epsilon$, then STOP (A is singular);
      assign $l$ position $k$ and remove $l$ from spike-index queue;
      rescale the pivot column so that the resulting pivot equals 1.
```

5.7 Example:

In this Section, we apply the sparse factorization algorithm described above to the 8×8 matrix introduced in chapter 4. The pivot tolerance τ is set to 0.50. The numerical values of the matrix and of its factors are given with a precision of

10^{-2}. Phases 1, 2 and 3 correspond to prescaling, preordering and factorization with restricted pivoting respectively. SI queue and SP queue denote the spike-index queue and the spike-position queue respectively. The columns belonging to the SI queue are printed in italic.

	1	2	3	4	5	6	7	8	count	score
1	8.00	9.00	5.00	1.00				-7.00	5	
2		3.00				8.00	2.00		3	
3				6.00	4.00				2	
4	-3.00	5.00		-9.00				2.00	4	
5		4.00	-7.00	3.00					3	
6					-2.00	5.00	-1.00		3	
7				5.00	-8.00	2.00	6.00	-4.00	5	
8		5.00	2.00					3.00	3	
count	2	5	3	5	3	3	3	4		
score										
scale	1 00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		

Matrix 0

Phase 1

The columns of the matrix are prescaled.

	1	2	3	4	5	6	7	8	count	score
1	**1.00**	1.00	-0.71	-0.11				1.00	5	
2		0.33				1.00	0.33		3	
3				-0.67	-0.50				2	
4	**-0.38**	0.56		1.00				-0.29	4	
5		0.44	1.00	-0.33					3	
6					0.25	0.63	-0.17		3	
7				-0.56	1.00	0.25	1.00	0.57	5	
8		0.56	-0.29					-0.43	3	
count	**2**	5	3	5	3	3	3	4		
score										
scale	8.00	9.00	-7.00	-9.00	-8.00	8.00	6.00	-7.00		

Matrix 1d

Phase 2 BOTTOMRIGHT1

Column 1 is selected and assigned position $p = 9 - 2 = 7$.

Rows 1 (pivot row) and 4 are selected and assigned positions 7 and 8.

Position 8 is sent to the SP queue.

	2	3	4	5	6	7	1	8	count	score
2	0.33				1.00	0.33			3	
3			-0.67	-0.50					2	
5	0.44	1.00	-0.33						3	
6				0.25	0.63	-0.17			3	
7			-0.56	1.00	0.25	1.00		0.57	5	
8	0.56	-0.29						-0.43	3	
1	1.00	-0.71	-0.11				1.00	1.00		
4	0.56		1.00				-0.38	-0.29		
count	3	1	3	3	3	3		2		
score										
scale	9.00	-7.00	-9.00	-8.00	8.00	6.00	8.00	-7.00		

Matrix 2d

Phase 2 TOPLEFT1

Row 3 is selected and assigned position 1.

Columns 4 and 5 are selected.

Column 4 (pivot column) is assigned position 1.

Column 5 is sent to the SI queue.

Column 4 is rescaled so that the pivot equals 1.

	4	2	3	6	7	8	1	5	count	score
3	1.00							*-0.50*		
2		0.33		1.00	0.33				3	
5	0.50	0.44	1.00						2	
6				0.63	-0.17			*0.25*	2	
7	0.83			0.25	1.00	0.57		*1.00*	3	
8		0.56	-0.29			-0.43			3	
1	0.17	1.00	-0.71			1.00	1.00			
4	-1.50	0.56				-0.29	-0.38			
count		3	2	3	3	2				
score			5.00			3.43				
scale	6.00	9.00	-7.00	8.00	6.00	-7.00	8.00	*-8.00*		

Matrix 3d

Phase 2 BOTTOMRIGHT2

Column 3 is selected and assigned position $p = 7 - 2 = 5$.

Rows 5 (pivot row) and 8 are selected and assigned positions 5 and 6.

Position 6 is sent to the SP queue.

	4	2	6	7	3	8	1	5	count	score
3	1.00							*-0.50*		
2		0.33	1.00	0.33					3	
6			**0.63**	**-0.17**				*0.25*	**2**	
7	0.83		0.25	1.00		0.57		*1.00*	3	
5	0.50	0.44			1.00					
8		0.56			-0.29	-0.43				
1	0.17	1.00			-0.71	1.00	1.00			
4	-1.50	0.56				-0.29	-0.38			
count		1	3	3		1				
score										
scale	6.00	9.00	8.00	6.00	-7.00	-7.00	8.00	*-8.00*		

Matrix 4d

Phase 2 TOPLEFT2

Row 6 is selected and assigned position 2.

Columns 6 and 7 are selected.

Column 6 (pivot column) is assigned position 2.

Column 7 is sent to the SI queue.

Column 6 is rescaled so that the pivot equals 1.

	4	6	2	8	3	*7*	1	*5*	count	score
3	1.00							*-0.50*		
6		1.00				*-0.17*		*0.25*		
2		1.60	0.33			*0.33*			1	
7	0.83	0.40		**0.57**		*1.00*		*1.00*	1	
5	0.50		0.44		1.00					
8			0.56	-0.43	-0.29					
1	0.17		1.00	1.00	-0.71		1.00			
4	-1.50		0.56	-0.29			-0.38			
count			1	1						
score			0.33	**0.57**						
scale	6.00	5.00	9.00	-7.00	-7.00	*6.00*	8.00	*-8.00*		

Matrix 5d

Phase 2 BOTTOMRIGHT3

Column 8 is selected and assigned position $p = 5 - 1 = 4$.

Row 7 (pivot row) is selected and assigned position 5.

Column 8 is rescaled so that the pivot equals 1.

	4	6	2	8	3	7	1	5	count	score
3	1.00							*-0.50*		
6		1.00				*-0.17*		*0.25*		
2		1.60	$\boxed{0.33}$			*0.33*			1	0
7	0.83	0.40		1.00		*1.00*		*1.00*		
5	0.50		0.44		1.00					
8			0.56	-0.75	-0.29					
1	0.17		1.00	1.75	-0.71		1.00			
4	-1.50		0.56	-0.50			-0.38			
count			1							
score										
scale	6.00	5.00	9.00	-4.00	-7.00	*6.00*	8.00	*-8.00*		

Matrix 6d

Phase 2 TOPLEFT3

Row 2 is selected and rejected because $|0.33| < \tau$.

Position 3 is sent to the SP queue.

Column 2 is sent to the SI queue.

Phase 3 Computation of $\overline{\boldsymbol{\sigma}}_3$

We remove the leftmost position, namely 3, from the spike-position queue. Then we have to solve the system $\boldsymbol{\omega}_3 \mathbf{A}_1 = \mathbf{u}_3^T$, or rather

$$\overline{\boldsymbol{\sigma}}_3 \begin{pmatrix} 1.00 & & \\ & 1.00 & \\ & 1.60 & 1.00 \end{pmatrix} = (\,0 \quad 0 \quad 1\,).$$

The solution, $\overline{\boldsymbol{\sigma}}_3 = (\,0.00 \quad -1.60 \quad 1.00\,)$, can be used to compute the pivots that would result from assigning columns 2, 7 and 5 position 3

$$(\,0.00 \quad -1.60 \quad 1.00\,) \begin{pmatrix} & & -0.50 \\ & -0.17 & 0.25 \\ 0.33 & 0.33 & \end{pmatrix} = (\,0.33 \quad 0.60 \quad -0.40\,).$$

Selecting column 7 yields the pivot of largest absolute value. Therefore, we remove column 7 from the spike-index queue and assigned it position 3. Then we rescale column 7 by 0.60 so that the resulting pivot equals 1.

	4	6	7	8	3	*2*	1	*5*
3	1.00							*-0.50*
6		1.00	-0.28					*0.25*
2		1.60	0.56			*0.33*		
7	0.83	0.40	1.67	1.00				*1.00*
5	0.50				1.00	*0.44*		
8				-0.75	-0.29	*0.56*		
1	0.17			1.75	-0.71	*1.00*	1.00	
4	-1.50			-0.50		*0.56*	-0.38	
scale	6.00	5.00	3.60	-4.00	-7.00	*9.00*	8.00	*-8.00*
omega	0.00	-1.60	1.00					
beta			0.60			*0.33*		*0.40*

Matrix 7d

Phase 3 Computation of $\overline{\boldsymbol{\sigma}}_6$

We remove the leftmost position, 6, from the spike-position queue. Then we have to solve the system $\boldsymbol{\omega}_6 \mathbf{E}_3 \mathbf{A}_1 = \mathbf{u}_3^T$, or rather

$$\overline{\boldsymbol{\sigma}}_6 \begin{pmatrix} 1.00 & & & & & \\ & 1.00 & -0.28 & & & \\ & 1.60 & 0.56 & & & \\ 0.83 & 0.40 & 1.67 & 1.00 & & \\ 0.50 & & & & 1.00 & \\ & & & -0.75 & -0.29 & 1.00 \end{pmatrix} = (\,0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1\,).$$

The solution, $\overline{\boldsymbol{\sigma}}_6 = (\,-0.77 \quad 1.83 \quad -1.33 \quad 0.75 \quad 0.29 \quad 1.00\,)$, can be used to compute the pivots that would result from assigning columns 5 and 2 position 6:

$$(\,-0.77 \quad 1.83 \quad -1.33 \quad 0.75 \quad 0.29 \quad 1.00\,) \begin{pmatrix} -0.50 & \\ 0.25 & \\ & 0.33 \\ 1.00 & \\ & 0.44 \\ & 0.56 \end{pmatrix} = (\,1.59 \quad 0.24\,).$$

Column 5 provides the pivot of largest absolute value. Therefore, we remove column 5 from spike-index queue and assign it position 6. Then we rescale it by

1.59 so that the resulting pivot equals 1.

	4	6	7	8	3	5	1	2
3	1.00					-0.31		
6		1.00	-0.28			0.16		
2		1.60	0.56					*0.33*
7	0.83	0.40	1.67	1.00		0.63		
5	0.50				1.00			*0.44*
8				-0.75	-0.29			*0.56*
1	0.17			1.75	-0.71		1.00	*1.00*
4	-1.50			-0.50			-0.38	*0.56*
scale	6.00	5.00	3.60	-4.00	-7.00	-12.7	8.00	*9.00*
omega	-0.77	1.83	-1.33	0.75	0.29	1.00		
beta						1.59		*0.24*

Matrix 8d

Phase 3 Computation of $\bar{\boldsymbol{\sigma}}_8$

We match the remaining spike-index 2 and spike-position 8. However, we must still solve $\boldsymbol{\omega}_8\mathbf{E}_6\mathbf{E}_3\mathbf{A}_1 = \mathbf{u}_3^T$ or equivalently

$$\bar{\boldsymbol{\sigma}}_8 \begin{pmatrix} 1.00 & & & & & -0.31 & & \\ & 1.00 & -0.28 & & & 0.16 & & \\ & 1.60 & 0.56 & & & & & \\ 0.83 & 0.40 & 1.67 & 1.00 & & 0.63 & & \\ 0.50 & & & & 1.00 & & & \\ & & & -0.75 & -0.29 & & & \\ 0.17 & & & 1.75 & -0.71 & & 1.00 & \\ -1.50 & & & -0.50 & & & -0.38 & 1.00 \end{pmatrix}$$

$$= (0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1)\,.$$

The solution is $\bar{\boldsymbol{\sigma}}_8 = (0.97 \quad 0.73 \quad -0.53 \quad 0.30 \quad 0.44 \quad 0.61 \quad 0.38 \quad 1.00)$ and the pivot resulting from placing column 2 in position 8 is given by

$$(0.97 \quad 0.73 \quad -0.53 \quad 0.30 \quad 0.44 \quad 0.61 \quad 0.38 \quad 1.00)\begin{pmatrix} \\ \\ 0.33 \\ \\ 0.44 \\ 0.56 \\ 1.00 \\ 0.56 \end{pmatrix} = 1.29\,.$$

Therefore, we rescale column 2 by 1.29 so that the pivot equals 1.

	4	6	7	8	3	5	1	2
3	1.00					-0.31		
6		1.00	-0.28			0.16		
2		1.60	0.56					0.20
7	0.83	0.40	1.67	1.00		0.63		
5	0.50				1.00			0.35
8				-0.75	-0.29			0.43
1	0.17			1.75	-0.71		1.00	0.78
4	-1.50			-0.50			-0.38	0.43
scale	6.00	5.00	3.60	-4.00	-7.00	-12.7	8.00	11.58
omega	0.97	0.73	-0.53	0.30	0.44	0.61	0.38	1.00
beta								1.29

Matrix 9d

The final matrix has three spikes, namely columns 7, 5 and 2, in positions 3, 6 and 8 respectively. Its factorization is given by the three vectors

$$\overline{\boldsymbol{\sigma}}_3 = (\,0.00 \quad -1.60 \quad 1.00\,)$$

$$\overline{\boldsymbol{\sigma}}_6 = (\,-0.77 \quad 1.83 \quad -1.33 \quad 0.75 \quad 0.29 \quad 1.00\,)$$

$$\overline{\boldsymbol{\sigma}}_8 = (\,0.97 \quad 0.73 \quad -0.53 \quad 0.30 \quad 0.44 \quad 0.61 \quad 0.38 \quad 1.00\,).$$

5.8 Block Triangular Reduction

Dulmage and Mendelsohn (1963) have indicated a procedure to permute the rows and the columns of a matrix so that the resulting matrix is lower triangular by blocks.

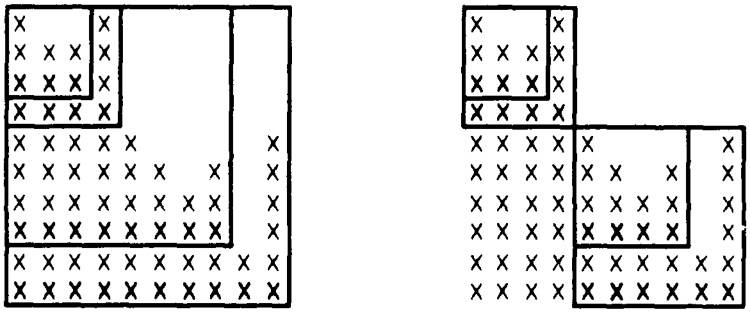
First, find a transversal, i.e. a permutation of the columns with matrix representation **Q** such that the diagonal of the resulting matrix **AQ** has no zero entry.

Then identify the resulting matrix **AQ** with its canonically associated directed graph (the nodes are the indices $\{1, \ldots, m\}$ and the arcs are the pairs (i, j) such that $A_{i,j} \neq 0$).

Finally determine the strongly connected components of the graph (cf. Tarjan, 1972), regroup the nodes of the same components into supernodes and consider the resulting collapsed graph: since it does not contain any cycle, the supernodes can be reordered so that if arc $I \rightarrow J$ exists then $I > J$. List the nodes from the first supernode, then those from the second one, and so on that list defines a row permutation matrix $\mathbf{P}$. The matrix $\mathbf{PAQP}^T$ is lower triangular by blocks.

Once the blocks have been identified, they can be reordered so as to minimize the number of spikes within the block and systems can be solved block by block. This reduces the size of the blocks to be reordered and the length of the auxiliary vectors $\boldsymbol{\omega}_k$ or $\boldsymbol{\sigma}_k$.

Example:



On the 10 × 10 matrix shown above, factorizing the diagonal blocks instead of the whole matrix reduces the number of nonzero components for the auxiliary vectors $\boldsymbol{\omega}_k$ or $\boldsymbol{\sigma}_k$ from 25 to 17.

CHAPTER 6. APPLICATION TO THE SIMPLEX ALGORITHM

6.1 Introduction

In order to test our factorization method, we have implemented it within the framework of the simplex algorithm for linear programming (Dantzig, 1963). As a reference code, we have chosen the FORTRAN optimization code MINOS 5.3 (Murtagh and Saunders, 1987) whose modularity, robustness and performance on large-scale problems make it an ideal benchmark.

In MINOS 5.3, the factorization of the basis, the update of the basis and the solution of linear systems involving the basis are carried out by a set of FORTRAN subroutines constituting the file MI25BFAC. We have written a file of FORTRAN subroutines, called MI26BFAC, designed to perform the same tasks using our method. We have run MINOS 5.3 alternatively with the original file MI25BFAC and with the new file MI26BFAC on different test-problems under different options.

6.2 The File MI26BFAC

The file MI26BFAC is designed to perform the same tasks as MI25BFAC when used within MINOS 5.3 to solve linear programming problems. The major implementation differences are the following. First, our factorization is inherently different from the LU factorization. Second, we compute a block-triangularization of the basis and then perform the factorization only on the diagonal blocks. Third, we use a conventional product-form update (with an in-core "η-file").

The principal subroutines of MI26BFAC, written in FORTRAN77, are represented hierarchically in Chart **6.2**. Their names and their roles are listed below:

`M2BFAC`: Performs the factorization of the basic matrix **B** by calling `M2BELM`, `M2BMAP`, `M2BSOL` and `M2SING`.

`M2BELM`: Extracts a sequence of triplets representing the matrix **B** from the data.

`POINTR`: Converts the representation of **B** from a sequence of triplets to sparse column format with an array of coefficient values, an array of row indices and an array of column pointers.

M2BMAP: Allocates storage for most of the arrays used in the factorization and updating of **B**.

M2BSOL: Performs one of the following actions according to the value of the parameter "mode":

factorize **B** by calling POINTR, TRIANG and TRTBLK;

solve $\mathbf{B}\mathbf{H}_1 \ldots \mathbf{H}_p\mathbf{P}\mathbf{x} = \mathbf{b}$ by calling SOLVEM, PERMCP and UPDATE;

solve $\boldsymbol{\pi}\mathbf{B}\mathbf{H}_1 \ldots \mathbf{H}_p = \boldsymbol{\gamma}\mathbf{P}^{-1}$ by calling SOLVEN, PERMCP and UPDATE;

add an element $\mathbf{H}_q$ to the η-file by calling ADDETA.

M2SING: Replaces one column of **B** by an appropriate slack column when the original matrix results in too small a pivot during factorization. It may be called several times in a row.

TRIANG: Identifies the lower block triangular representation of **B** by calling TRANSV and GETBLK.

TRTBLK: Extracts, prescales, preorders and factorizes each diagonal block of **B** by calling EXTRCT, RESCAL, PREORD and FACTOR.

SOLVEM: Solves $\mathbf{B}\mathbf{x} = \mathbf{b}$ with the help of SOLVEA.

SOLVEN: Solves $\boldsymbol{\pi}\mathbf{B} = \boldsymbol{\gamma}$ with the help of SOLVEB.

PERMCP: Permutes a vector according to $\mathbf{P}$ or $\mathbf{P}^{-1}$.

UPDATE: Performs rank-one updates according to the η-file.

ADDETA: Adds a column-vector characterizing a factor $\mathbf{H}_q$ to the η-file.

TRANSV: Computes a transversal of **B**.

GETBLK: Computes the diagonal blocks of **B**.

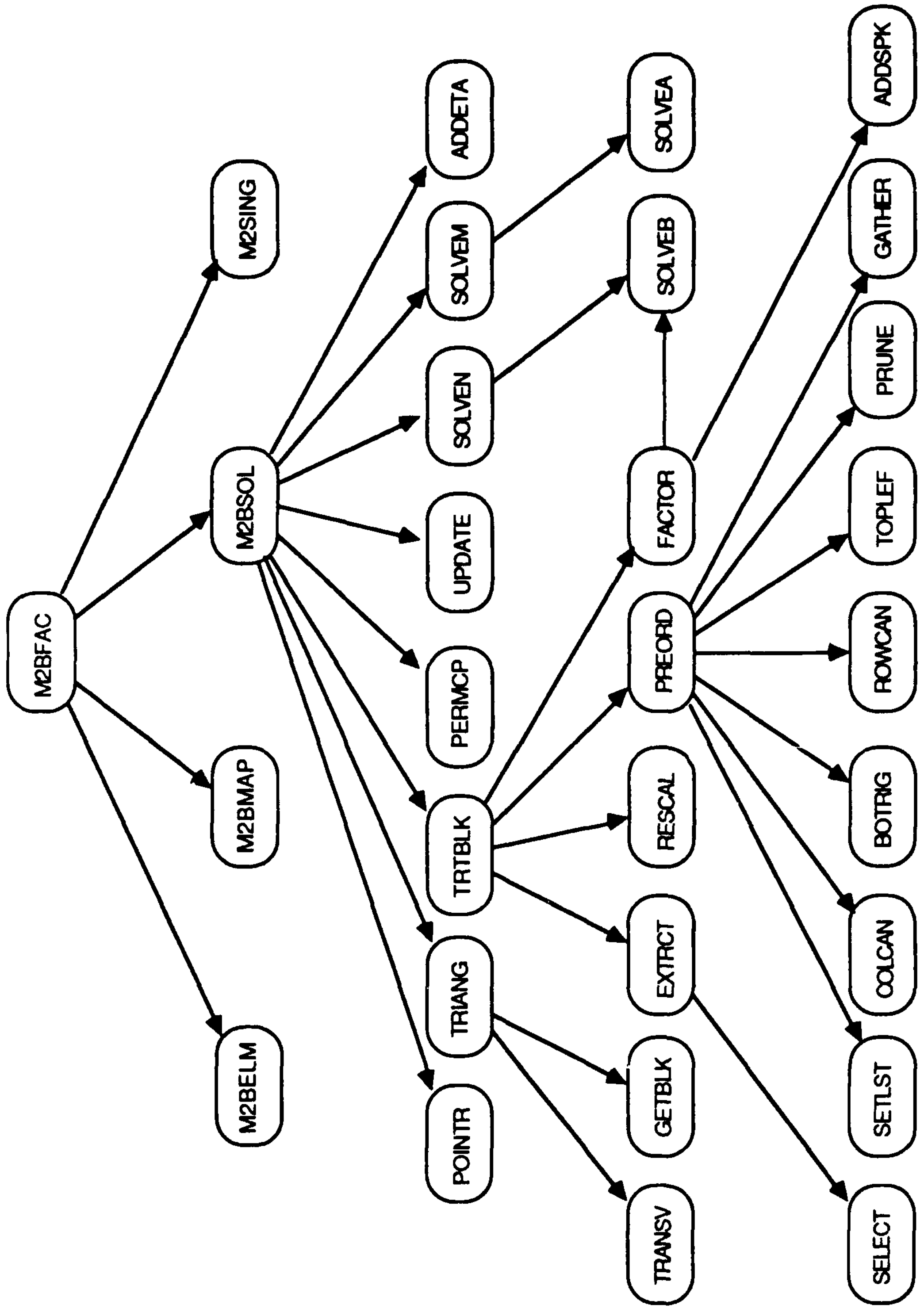
EXTRCT: Identifies the elements from a diagonal block of **B** with the help of SELECT.

SELECT: Scans the columns that support a diagonal block to identify the entries that belong to that block.

RESCAL: Normalizes the columns of each block.

PREORD: Applies a spike reduction algorithm to preorder the blocks by calling SETLST, COLCAN, BOTRIG, ROWCAN, TOPLEF, PRUNE and GATHER.

Structure of MI26BFAC



FACTOR: Factorizes the blocks by calling SOLVEB and ADDSPK. Sorts the column representation of $\mathbf{B}$ according to the final row order.

SETLST: Sets up a row-wise linked list representing a diagonal block and some auxiliary variables for preordering.

COLCAN: Finds column candidates to be used by BOTRIG.

BOTRIG: Performs a BottomRight iteration.

ROWCAN: Finds row candidates to be used by TOPLEF.

TOPLEF: Performs a TopLeft iteration.

PRUNE : Performs the deletion of entries in row-wise linked lists.

GATHER: Gathers and orders all spikes in order of non-increasing height.

SOLVEA: Solves $\mathbf{B}'\mathbf{x}' = \mathbf{b}'$ where $\mathbf{B}'$ is a sub-block of $\mathbf{B}$.

SOLVEB: Solves $\boldsymbol{\pi}'\mathbf{B}' = \boldsymbol{\gamma}'$ where $\mathbf{B}'$ is a sub-block of $\mathbf{B}$.

ADDSPK: Adds one auxiliary vector $\boldsymbol{\sigma}_k$ to the factor list.

6.3 Test Implementation

We have run MINOS 5.3 on a SUN 3/50 workstation using a 16 Mhz Motorola 68020 CPU under the SunOS 3.5 operating system and with the Berkeley f77 compiler.

We have chosen a set of 53 test-problems studied by Lustig (1987) and made publicly available on *netlib* (Dongarra and Grosse, 1987) by Gay (1985). Although the original MINOS code could run all these test-problems without modification, our implementation required too much storage. Therefore, we have increased the size of the array Z from 100000 to 120000 for SHIP12L and to 150000 for PILOTJA and decided to forego the last two problems 80BAU3B and PILOT because it was apparent from running the other test-problems that our method would be unpractical and widely outperformed by MINOS.

With each of the 51 other test-problems, we have made two experiments, each based on a different MINOS option file.

6.4 First Experiment

The first experiment is the straightforward solve of each linear program from scratch in order to compare overall performance. The option file is the following:

```
ROWS                  1500
COLUMNS               5500
ELEMENTS             22000
MPS FILE                10
NEW BASIS FILE          12
SAVE FREQUENCY       10000
ITERATION LIMIT      20000
PRINT LEVEL              0
SOLUTION                NO
SCALE OPTION             2
PARTIAL PRICE           10
FACTOR FREQUENCY        25
LU FACTOR TOLERANCE  100.0
LU UPDATE TOLERANCE   10.0
```

In Table **6.4.1**, we have listed for each test-problem the number of iterations l_v in phase 1, the total number of iterations k_v and the execution time t_v under both versions ($v = 0$ for MI25BFAC and $v = 1$ for MI26BFAC), as well as the ratio of execution times t_1/t_0. In 19 cases, our implementation runs faster than MINOS, which is hampered by the factorization frequency of 25.

In Table **6.4.2**, we have listed for each test-problem the optimal objective value z_v under both versions ($v = 0$ for MI25BFAC and $v = 1$ for MI26BFAC), as well as their relative difference. In 44 instances, the optimal objective values agree up to 11 significant digits. The largest discrepancy occurs with PILOTWE where the agreement is still of 6 significant digits. This seems to indicate that our method achieves a satisfactory numerical stability.

TABLE 6.4.1	l1	l0	k1	k0	t1	t0	t1/t0
AFIRO	3	3	7	7	1.94	1.84	1.05
ADLITTLE	23	23	117	117	8.00	7.18	1.11
SC205	0	0	145	145	35.58	17.84	1.99
SCAGR7	79	79	99	99	11.50	10.14	1.13
SHARE2B	67	67	110	110	10.52	9.48	1.11
RECIPE	7	7	33	33	5.88	6.30	0.93
VTPBASE	33	33	48	48	10.10	9.72	1.04
SHARE1B	135	135	261	261	27.86	23.56	1.18
BORE3D	114	114	160	160	27.38	25.46	1.08
SCORPION	63	63	102	102	23.40	24.88	0.94
CAPRI	164	164	251	251	41.32	37.72	1.10
SCAGR25	169	172	353	376	81.88	86.00	0.95
SCTAP1	195	195	287	287	46.78	47.72	0.98
BRANDY	269	269	423	423	80.42	65.66	1.22
ISRAEL	44	44	250	250	49.28	37.18	1.33
ETAMACRO	313	302	649	574	129.30	109.30	1.18
SCFXM1	222	222	389	389	70.54	68.98	1.02
GROW7	0	0	232	232	54.86	49.82	1.10
BANDM	225	225	498	498	114.10	101.06	1.13
E226	111	111	480	467	81.60	67.76	1.20
STANDATA	40	40	114	111	31.00	31.10	1.00
SCSD1	92	92	368	370	40.88	37.68	1.08
GFRDPNC	291	283	615	637	143.02	[illegible]7.26	0.86
BEACONFD	39	39	104	87	25.62	22.82	1.12
STAIR	334	334	449	449	385.68	197.58	1.95
SCRS8	63	63	692	645	176.42	161.68	1.09
SEBA	225	225	399	399	112.00	106.14	1.06
SHELL	60	60	274	274	73.26	82.02	0.89
PILOT4	445	445	1599	1467	1278.24	550.82	2.32
SCFXM2	538	538	819	819	259.50	270.40	0.96
SCSD6	206	206	1139	1139	172.96	148.24	1.17
GROW15	0	0	485	485	226.32	196.24	1.15
SHIP04S	13	13	158	159	47.12	54.42	0.87
FFFFF800	883	806	1074	1002	292.22	284.08	1.03
GANGES	412	410	700	705	333.02	354.66	0.94
SCFXM3	828	854	1349	1391	577.54	660.18	0.87
SCTAP2	362	373	766	744	319.02	346.84	0.92
GROW22	0	0	736	662	571.92	369.46	1.55
SHIP04L	13	13	275	276	77.60	88.86	0.87
PILOTWE	380	492	4474	4040	7900.80	1938.60	4.08
SIERRA	470	452	1083	1071	438.18	511.84	0.86
SHIP08S	17	17	262	262	106.68	130.24	0.82
SCTAP3	492	425	926	883	488.96	546.74	0.89
SHIP12S	39	39	434	434	196.14	273.56	0.72
25FV47	2046	2304	7821	7828	10704.38	4524.18	2.37
SCSD8	767	711	3102	3914	1383.04	1245.60	1.11
NESM	1176	1251	3270	3399	1357.34	1189.00	1.14

TABLE 6.4.2	z1	z0	z1-z0/z0
AFIRO	-4.6475314286e+02	-4.6475314286e+02	0.00e+00
ADLITTLE	2.2549496316e+05	2.2549496316e+05	0.00e+00
SC205	-5.2202061212e+01	-5.2202061212e+01	0.00e+00
SCAGR7	-2.3313892548e+06	-2.3313892548e+06	0.00e+00
SHARE2B	-4.1573224074e+02	-4.1573224074e+02	0.00e+00
RECIPE	-2.6661600000e+02	-2.6661600000e+02	0.00e+00
VTPBASE	1.2983146246e+05	1.2983146246e+05	0.00e+00
SHARE1B	-7.6589318579e+04	-7.6589318579e+04	0.00e+00
BORE3D	1.3730803942e+03	1.3730803942e+03	0.00e+00
SCORPION	1.8781248227e+03	1.8781248227e+03	0.00e+00
CAPRI	2.6900129138e+03	2.6900129138e+03	0.00e+00
SCAGR25	-1.4753433061e+07	-1.4753433061e+07	0.00e+00
SCTAP1	1.4122500000e+03	1.4122500000e+03	0.00e+00
BRANDY	1.5185098965e+03	1.5185098965e+03	0.00e+00
ISRAEL	-8.9664482186e+05	-8.9664482186e+05	0.00e+00
ETAMACRO	-7.5571521819e+02	-7.5571521831e+02	-1.59e-10
SCFXM1	1.8416759028e+04	1.8416759028e+04	0.00e+00
GROW7	-4.7787811815e+07	-4.7787811815e+07	0.00e+00
BANDM	-1.5862801845e+02	-1.5862801845e+02	0.00e+00
E226	-1.8751929066e+01	-1.8751929066e+01	0.00e+00
STANDATA	1.2576995000e+03	1.2576995000e+03	0.00e+00
SCSD1	8.6666666743e+00	8.6666666743e+00	0.00e+00
GFRDPNC	6.9022359995e+06	6.9022359995e+06	0.00e+00
BEACONFD	3.3592485807e+04	3.3592485807e+04	0.00e+00
STAIR	-2.5126695119e+02	-2.5126695119e+02	0.00e+00
SCRS8	9.0429998619e+02	9.0429998619e+02	0.00e+00
SEBA	1.5711600000e+04	1.5711600000e+04	0.00e+00
SHELL	1.2088253460e+09	1.2088253460e+09	0.00e+00
PILOT4	-2.5811392641e+03	-2.5811392641e+03	0.00e+00
SCFXM2	3.6660261565e+04	3.6660261565e+04	0.00e+00
SCSD6	5.0500000078e+01	5.0500000078e+01	0.00e+00
GROW15	-1.0687094129e+08	-1.0687094129e+08	0.00e+00
SHIP04S	1.7987147004e+06	1.7987147004e+06	0.00e+00
FFFFF800	5.5567961219e+05	5.5567959103e+05	3.81e-08
GANGES	-1.0958634770e+05	-1.0958635225e+05	-4.15e-08
SCFXM3	5.4901254550e+04	5.4901254550e+04	0.00e+00
SCTAP2	1.7248071429e+03	1.7248071429e+03	0.00e+00
GROW22	1.6083433648e+08	1.6083433648e+08	0.00e+00
SHIP04L	1.7933245380e+06	1.7933245380e+06	0.00e+00
PILOTWE	-2.7200991196e+06	-2.7201034525e+06	-1.59e-06
SIERRA	1.5394362184e+07	1.5394362184e+07	0.00e+00
SHIP08S	1.9200982105e+06	1.9200982105e+06	0.00e+00
SCTAP3	1.4240000000e+03	1.4240000000e+03	0.00e+00
SHIP12S	1.4892361344e+06	1.4892361344e+06	0.00e+00
25FV47	5.5018467791e+03	5.5018458883e+03	1.62e-07
SCSD8	9.0499999993e+02	9.0499999993e+02	0.00e+00
NESM	1.4076073324e+07	1.4076075128e+07	-1.28e-07
CZPROB	2.1851966989e+06	2.1851966989e+06	0.00e+00
PILOTJA	-6.1131152948e+03	-6.1131157775e+03	-7.90e-08
SHIP08L	1.9090552114e+06	1.9090552114e+06	0.00e+00
SHIP12L	1.4701879193e+06	1.4701879193e+06	0.00e+00

6.5 Second Experiment

The second experiment is designed to focus on the factorization size. As initial basis, we use the optimal basis obtained during the first experiment under MINOS with MI25BFAC. The objective function is now maximized instead of minimized and the iteration limit is set to 25. The option file is the following:

```
ROWS                   1500
COLUMNS                5500
ELEMENTS              22000
MAXIMIZE
MPS FILE                 10
OLD BASIS FILE           12
NEW BASIS FILE            0
SAVE FREQUENCY        10000
ITERATION LIMIT          25
PRINT LEVEL               1
SOLUTION                 NO
SCALE OPTION              2
PARTIAL PRICE            10
FACTOR FREQUENCY        100
LU FACTOR TOLERANCE   100.0
LU UPDATE TOLERANCE    10.0
```

Not surprisingly, some of the test-runs end with an unbounded solution. In four other instances, MINOS lists some variables as apt to increase indefinitely. These occurences are indicated respectively by a U and an I in the first column of Table **6.5.1**.

Table **6.5.1** also contains the number of iterations k up to which both methods yield the same basic solution, and the total number of iterations k'. For each test-problem, that number, usually the iteration limit of 25, turns out to be the same under both versions.

Finally, Table **6.5.1** provides the size ∂ of the largest diagonal block, the

TABLE 6.5.1		k	k'	∂	f1	f0	f1/f0	t1	t0	t1/t0
AFIRO		8	8	4	0.04	0.04	1.00	1.70	1.98	0.86
ADLITTLE		25	25	13	0.06	0.14	0.43	4.32	4.22	1.02
SC205		20	25	184	0.78	0.62	1.26	11.06	7.38	1.50
SCAGR7		25	25	0	0.04	0.18	0.22	5.72	5.50	1.04
SHARE2B		25	25	10	0.10	0.24	0.42	5.70	5.34	1.07
RECIPE		14	25	0	0.02	0.18	0.11	5.40	6.32	0.85
VTPBASE	U	22	22	31	0.10	0.36	0.28	7.88	7.84	1.01
SHARE1B		25	25	10	0.10	0.34	0.29	7.62	7.50	1.02
BORE3D	U	24	24	44	0.22	0.54	0.41	11.52	11.26	1.02
SCORPION		25	25	14	0.32	1.00	0.32	12.28	12.96	0.95
CAPRI		25	25	44	0.24	0.70	0.34	13.22	11.90	1.11
SCAGR25		25	25	54	0.28	0.92	0.30	16.14	15.30	1.05
SCTAP1		25	25	6	0.14	0.50	0.28	13.28	12.86	1.03
BRANDY		25	25	90	0.60	0.90	0.67	14.52	13.12	1.11
ISRAEL	U	0	0	54	0.22	0.60	0.37	8.94	9.24	0.97
ETAMACRO		25	25	50	0.28	0.92	0.30	17.52	17.50	1.00
SCFXM1		25	25	13	0.24	0.80	0.30	15.60	15.42	1.01
GROW7		25	25	115	1.40	1.46	0.96	18.04	15.44	1.17
BANDM		25	25	83	0.58	1.22	0.48	17.34	16.80	1.06
E226		25	25	62	0.28	0.72	0.39	14.28	13.20	1.08
STANDATA	U	0	0	2	0.12	0.46	0.26	17.16	17.28	0.99
SCSD1		25	25	14	0.08	0.22	0.36	15.00	14.66	1.02
GFRDPNC		25	25	0	0.12	1.04	0.12	25.66	26.40	0.97
BEACONFD	U	1	1	2	0.08	0.38	0.21	12.06	12.06	1.00
STAIR		25	25	324	12.36	4.50	2.75	47.60	27.48	1.73
SCRS8		25	25	23	0.34	1.02	0.33	23.74	23.78	1.00
SEBA		25	25	0	0.14	1.26	0.11	27.54	27.06	1.02
SHELL	I	25	25	0	0.26	0.90	0.29	30.00	30.28	0.99
PILOT4		25	25	279	9.42	1.95	4.83	49.58	32.92	1.51
SCFXM2		25	25	25	0.58	1.80	0.32	32.10	33.08	0.97
SCSD6		24	25	30	0.18	0.42	0.43	25.00	25.18	0.99
GROW15		25	25	79	2.10	3.90	0.54	32.44	33.16	0.98
SHIP04S	I	25	25	4	0.16	1.10	0.15	25.52	26.68	0.96
FFFFF800		25	25	27	0.32	1.24	0.26	32.04	31.28	1.02
GANGES		24	25	32	0.98	3.72	0.26	45.10	44.70	1.01
SCFXM3		25	25	25	0.86	2.92	0.29	45.58	47.38	0.96
SCTAP2		0	25	5	0.38	2.02	0.19	45.04	46.96	0.96
GROW22		0	25	119	4.08	6.02	0.68	47.18	45.82	1.03
SHIP04L	I	25	25	0	0.18	0.68	0.26	33.50	35.88	0.93
PILOTWE		25	25	519	15.70	4.3	3.65	87.64	58.70	1.49
SIERRA		25	25	0	0.22	2.02	0.11	61.34	62.80	0.98
SHIP08S	U	18	18	8	0.24	2.26	0.11	40.44	42.90	0.94
SCTAP3	I	2	25	12	0.40	2.94	0.14	58.28	59.80	0.97
SHIP12S	U	1	1	5	0.36	3.76	0.10	42.14	45.92	0.92
25FV47		25	25	365	6.82	4.44	1.54	73.02	54.72	1.33
SCSD8		23	25	209	1.14	1.52	0.75	52.32	51.30	1.02
NESM		25	25	179	1.08	1.88	0.57	75.24	73.22	1.03
CZPROB		25	25	6	0.30	3.56	0.08	63.02	66.50	0.95
PILOTJA		25	25	544	25.86	5.96	4.34	120.98	70.98	1.70
SHIP08L		25	25	8	0.24	2.42	0.10	66.36	68.80	0.96
SHIP12L	U	1	25	5	0.30	3.56	0.08	72.54	76.08	0.95

TABLE 6.5.2	a	w	å+w	å+w+x	l	u	l+u	l'+u'	s1/s0	s'1/s'0
AFIRO	72	3	74	150	23	51	74	95	1.00	1.58
ADLITTLE	251	31	261	1029	103	162	265	466	0.98	2.21
SC205	543	991	1493	4837	296	514	810	1045	1.84	4.63
SCAGR7	477	0	477	1787	0	477	477	826	1.00	2.16
SHARE2B	522	52	529	1509	116	444	560	707	0.94	2.13
RECIPE	271	0	271	375	0	271	271	266	1.00	1.41
VTPBASE	637	49	673	1896	96	546	642	606	1.05	3.13
SHARE1B	586	72	599	2060	223	424	647	846	0.93	2.43
BORE3D	1053	67	1084	2505	270	818	1088	1505	1.00	1.66
SCORPION	1680	248	1750	2864	369	1383	1752	2160	1.00	1.33
CAPRI	1193	127	1291	3562	250	992	1242	1234	1.04	2.89
SCAGR25	1644	72	1706	5512	192	1497	1689	2240	1.01	2.46
SCTAP1	950	18	954	1876	200	752	952	1113	1.00	1.69
BRANDY	1320	896	2057	4577	691	844	1535	1949	1.34	2.35
ISRAEL	•	•	•	•	•	•	•	•	•	•
ETAMACRO	1321	135	1433	4153	253	1149	1402	1571	1.02	2.64
SCFXM1	1321	66	1347	3148	302	1065	1367	1594	0.99	1.97
GROW7	1751	1983	3089	6080	1025	1200	2225	3010	1.39	2.02
BANDM	1997	631	2527	6624	874	1364	2238	2846	1.13	2.33
E226	1313	292	1540	3424	551	883	1434	1662	1.07	2.06
STANDATA	•	•	•	•	•	•	•	•	•	•
SCSD1	289	32	306	1389	108	229	337	639	0.91	2.17
GFRDPNC	1815	0	1815	2714	0	1815	1815	2112	1.00	1.29
BEACONFD	1275	1	1275	•	2	1273	1275	•	1.00	•
STAIR	3588	10960	13539	21917	2467	2896	5363	7436	2.52	2.95
SCRS8	1536	73	1578	4974	214	1409	1623	2040	0.97	2.44
SEBA	2790	0	2790	4433	0	2790	2790	3014	1.00	1.47
SHELL	1493	0	1493	1894	0	1493	1493	1636	1.00	1.16
PILOT4	3298	1032	12480	20107	1941	2395	4336	5518	2.88	3.64
SCFXM2	2698	173	2792	3554	677	2131	2808	3109	0.99	1.14
SCSD6	532	75	588	2080	163	454	617	910	0.95	2.29
GROW15	4216	2742	5237	7173	2421	2837	5258	6137	1.00	1.17
SHIP04S	1411	2	1412	1697	320	1093	1413	1493	1.00	1.14
FFFFF800	2708	32	2725	4287	376	2340	2716	2895	1.00	1.48
GANGES	5614	395	5662	8583	672	4965	5637	5636	1.00	1.52
SCFXM3	4089	291	4259	7314	1082	3197	4279	4605	1.00	1.59
SCTAP2	3083	16	3086	3491	100	2982	3082	3331	1.00	1.05
GROW22	6344	4327	7972	10957	3813	4043	7856	8727	1.01	1.26
SHIP04L	1409	0	1409	1678	0	1409	1409	1221	1.00	1.37
PILOTWE	3068	18634	21331	34526	2187	2576	4763	5783	4.48	5.97
SIERRA	2968	0	2968	3420	0	2968	2968	3011	1.00	1.14
SHIP08S	2787	7	2791	3009	501	2290	2791	2870	1.00	1.05
SCTAP3	4164	16	4170	4662	175	3987	4162	4331	1.00	1.08
SHIP12S	4111	3	4111	•	867	3244	4111	•	1.00	•
25FV47	4670	10465	14666	27457	2295	3706	6001	6650	2.44	4.13
SCSD8	1546	992	2475	6443	521	1356	1877	2256	1.32	2.86
NESM	2225	986	3034	9927	395	2073	2468	2728	1.23	3.64
CZPROB	3595	5	3598	4157	887	2712	3599	3759	1.00	1.11
PILOTJA	5206	27209	31151	48335	3467	3859	7326	9084	4.25	5.32
SHIP08L	2790	7	2795	3162	490	2304	2794	2874	1.00	1.10
SHIP12L	4111	3	4111	•	866	3245	4111	•	1.00	•

factorization times f_v under both versions ($v = 0$ for MI25BFAC and $v = 1$ for MI26BFAC), as well as their ratio f_1/f_0, and the total running time t_v under both versions ($v = 0$ for MI25BFAC and $v = 1$ for MI26BFAC), as well as their ratio t_1/t_0. Our method outperforms MINOS in 44 of the factorization times but only in 21 of the total execution times.

In addition to the auxiliary vectors $\boldsymbol{\omega}_k$ (or $\boldsymbol{\sigma}_k$), our factorization requires the storage of some elements of the original basis. The only elements of the basis that need not be stored are located in the lower triangular part of the diagonal blocks. More precisely, the first k elements of the rows of index k where column k is a spike are not needed (once the vector $\boldsymbol{\omega}_k$ has been computed). We denote by a the number of nonzero elements in the original basis, by å the number of these nonzero elements that can be viewed as part of the factorization, by w the number of nonzero elements in the auxiliary vectors $\boldsymbol{\omega}_k$, by x the number of elements stored in the η-file. Thus the amount of storage required by our method is $s_1 = \text{å}+w$ for the original basis and $s_1' = \text{å}+w+x$ for the k^{th} basis.

Finally, we denote by l,u and l',u' the sizes of the LU factors computed by MINOS for the first and k^{th} basis respectively. The amount of storage required by MINOS is $s_0 = l + u$ for the original basis and $s_0' = l' + u'$ for the k^{th} basis.

In Table **6.5.2**, we have listed the values of a and w, the sizes s_1 and s_1' of the factorizations under MI26BFAC, the values l and u, the sizes s_0 and s_0' of the factorizations under MI25BFAC, as well as the ratios s_1/s_0 and s_1'/s_0'. In test-problems where unboundedness has been detected, some factorizations have not been carried out, rendering some of the above values unavailable. These instances are indicated by a $*$.

In terms of sparsity, our method performs as well as or slightly better than MINOS in 32 instances of direct factorizations. However it performs uniformly worse after 25 column updates.

6.6 Conclusion

The testing experiment of Section **6.4** indicates that our factorization method is numerically stable in practically all instances.

The testing experiment of Section **6.5** demonstrates that our factorization method is more efficient when the diagonal blocks computed during the block-trianguiarization are small, say of order 50 or less. This can be attributed to the following points:

- The identification of the diagonal blocks resulting from the block-triangularization is efficient.
- Only the diagonal blocks are factored.
- A sorting of the rows that speeds up the reordering of each block is implemented along with the factorization.

On the other hand, the same experiment shows that, when the block-triangularization yields a large diagonal block, the size of our factorization becomes prohibitively large and renders the whole method inefficient. In practically all instances the numerical stability remains satisfactory.

Unfortunately, there does not seem to exist an efficient updating algorithm for the block-triangularization of a matrix. Because our method only factorizes the diagonal blocks, this makes a conventional product-form updating almost mandatory. This method of updating appears less efficient than the Bartels-Golub updating of the LU factors implemented in MINOS 5.3.

In spite of these drawbacks, we hope that future computer architectures and numerical software will suggest more applications for all or part of our factorization method.

BIBLIOGRAPHY

[1] R.H. Bartels (1971). A stabilization of the simplex method, *Numerische Mathematik* 16, 414–434.

[2] R.H. Bartels and G.H. Golub (1969). The simplex method of linear programming using the LU decomposition, *Communications of the ACM* 12, 266–268.

[3] V. Chvatal (1983). *Linear Programming*, W.H. Freeman and Co., New York and San Francisco.

[4] G.B. Dantzig (1963). *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey.

[5] G.B. Dantzig (1963). Compact basis triangularization for the simplex method, in *Recent Advances in Mathematical Programming* (R.L. Graves and P. Wolfe, eds.), McGraw-Hill, New York, 125–132.

[6] G.B. Dantzig (1985). *Another Product Form of the Inverse*, unpublished manuscript.

[7] J.J. Dongarra and E. Grosse (1987). Distribution of mathematical software via electronic mail, *Communications of the ACM* 30, 403–407.

[8] I.S. Duff, A.M. Erisman and J.K. Reid (1986). *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford.

[9] A.L. Dulmage and N.S. Mendelsohn (1963). Two algorithms for bipartite graphs, *J. of SIAM* 11, 183–194.

[10] A.M. Erisman, R.G. Grimes, J.G. Lewis and W.G. Poole (1985). A structurally stable modification of Hellerman-Rarick's P^4 algorithm for reordering unsymmetric sparse matrices, *SIAM J. of Numerical Analysis* 22, 369–385.

[11] D.M. Gay (1985). Electronic mail distribution of linear programming test problems, *Mathematical Programming Society COAL Newsletter* 13, 10–12.

[12] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright (1984). Sparse matrix methods in optimization, *SIAM J. of Scientific and Statistical Computing* 5, 562–589.

[13] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright (1987). Maintaining LU factors of a general sparse matrix, *Linear Algebra and its Applications* 88/89, 239–270.

[14] P.E. Gill, W. Murray and M.H. Wright (1981). *Practical Optimization*, Academic Press, London and New York.

[15] G.H. Golub and C.F. Van Loan (1983). *Matrix Computations*, Johns Hopkins University Press.

[16] E. Hellerman and D.C. Rarick (1971). Reinversion with the preassigned pivot procedure, *Mathematical Programming* 1, 195–216.

[17] E. Hellerman and D.C. Rarick (1972). The partitioned preassigned pivot procedure (P^4), in *Sparse Matrices and Their Applications* (D.J. Rose and R.A. Willoughby, eds.), Plenum Press, New York, 67–76.

[18] I.J. Lustig (1987). An analysis of an available set of linear programming test problems, Report SOL 87-11, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California.

[19] B.A. Murtagh (1981). *Advanced Linear Programming*, McGraw-Hill, New York.

[20] B.A. Murtagh and M.A. Saunders (1978). Large-scale linearly constrained optimization, *Mathematical Programming* 14, 41–72.

[21] B.A. Murtagh and M.A. Saunders (1987). MINOS 5.1 User's Guide, Report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California.

[22] K.G. Murty (1976). *Linear and Combinatorial Programming*, John Wiley and Sons, New York.

[22] M.A. Saunders (1976). A fast, stable implementation of the simplex method using Bartels-Golub updating, in *Sparse Matrix Computations* (J.R. Bunch and D.J. Rose, eds.), Academic Press, New York, 213–226.

[23] G. Strang (1976). *Linear Algebra and its Applications*, Academic Press, London and New York.

[24] R. Tarjan (1972). Depth-first search and linear graph algorithms, *SIAM J. of Computing* 1, 146–160.

[25] M. Yannakakis (1981). Computing the minimum fill-in is NP-complete, *SIAM J. of Algebraic and Discrete Mathematics* 2, 77–79.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
SOL 89-10

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A MATRIX FACTORIZATION AND ITS APPLICATION TO LARGE-SCALE LINEAR PROGRAMMING

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Pierre F. de Mazancourt

8. CONTRACT OR GRANT NUMBER(s)
#N00014-89-J-1659

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Department of Operations Research - SOL
Stanford University
Stanford, CA 94305-4022

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
1111MA

11. CONTROLLING OFFICE NAME AND ADDRESS
Office of Naval Research - Dept. of the Navy
800 N. Quincy Street
Arlington, VA 22217

12. REPORT DATE
July 1989

13. NUMBER OF PAGES
81 pages

15. SECURITY CLASS. (of this report)
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
large-scale optimization; linear programming; matrix factorization; matrix ordering; simplex method; sparse matrix

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(see reverse side)

A MATRIX FACTORIZATION AND ITS APPLICATION TO LARGE-SCALE LINEAR PROGRAMMING

Pierre F. de Mazancourt

SOL 89-10 ABSTRACT

As an alternative to the LU matrix factorization, we consider a factorization that uses the lower triangular part of the original matrix as one factor and computes the other factors as a product of rank-one update matrices.

Under some non-singularity assumptions, an $m \times m$ matrix $\mathbf{A}$ can be factorized as $\mathbf{E}_m \mathbf{E}_{m-1} \ldots \mathbf{E}_2 \mathbf{A}_1$ where $\mathbf{A}_1$ is the lower triangular part of $\mathbf{A}$ and $\mathbf{E}_k$ is a rank-one update matrix of the form $\mathbf{I} + \mathbf{v}_k \boldsymbol{\omega}_k$ with $\mathbf{v}_k$ a column vector and $\boldsymbol{\omega}_k$ a row vector. The vector $\mathbf{v}_k$ is the k^{th} column of $\mathbf{A} - \mathbf{A}_1$. If $\mathbf{v}_k = \mathbf{0}$, then $\mathbf{E}_k = \mathbf{I}$ may be omitted from the factorization. Otherwise, the row vector $\boldsymbol{\omega}_k$ must be computed.

After reviewing and improving the time complexity, the requirements, the stability and the efficiency of this method, we derive a stable factorization algorithm which we implement in FORTRAN77 within the framework of the simplex algorithm for linear programming.

A comparison of our numerical results with those obtained through the code MINOS 5.3 indicate that our method may be more efficient than an ordinary LU decomposition for some matrices whose order ranges between 28 and 1481, especially when these matrices are almost triangular.